

# NAVAL POSTGRADUATE SCHOOL MONTEREY, CALIFORNIA



## THESIS

### DOCUMENTATION FOR CAPS USER INTERFACE AND GRAPHIC EDITOR

by

Antionette Carin Bell

March, 1997

Thesis Advisor:  
Co-Advisor:

Luqi  
Berzins

Approved for public release; distribution is unlimited

DTIC QUALITY INSPECTED 8

19970925 024

# REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE March 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis
----------------------------------	------------------------------	---

4. TITLE AND SUBTITLE DOCUMENTATION FOR CAPS USER INTERFACE AND GRAPHIC EDITOR	5. FUNDING NUMBERS
--	--------------------

6. AUTHOR(S) Antionette Carin Bell
------------------------------------

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
---	--

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
---	--

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE
---	------------------------

13. ABSTRACT (maximum 200 words)

The Computer-Aided Prototyping System (CAPS) is an integrated environment, comprised of an integrated set of software tools, aimed at rapidly prototyping hard real-time embedded systems. The problem with the current CAPS software development environment is the absence of a CAPS User's Manual (How to Use ...), which provides CAPS users with step-by-step guidelines on how to use the CAPS tools.

One solution to this problem was solved by designing, developing, and creating online documentation for the CAPS User Interface and Graphic Editor Reference Manuals for CAPS Release 1.1. Such an approach provides accessible visual, graphical, and textual step-by-step illustrations for CAPS users when interfacing, interacting, and manipulating, commands and options within the CAPS User Interface subsystem and Graphic Editor tool. In addition, this approach includes a glossary which helps CAPS users to understand the meaning of the difficult or specialized terms used in this environment.

14. SUBJECT TERMS user interface, syntax-directed editor, graphic editor, Prototyping System Description Language (PSDL), computational graph, data flow diagram, vertex, bubble	15. NUMBER OF PAGES 134
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL
---	--	---	----------------------------------



Approved for public release; distribution is unlimited

**DOCUMENTATION FOR CAPS USER INTERFACE AND GRAPHIC EDITOR**

Antionette Carin Bell  
Lieutenant, United States Navy  
B.S., Savannah State College, 1989

Submitted in partial fulfillment  
of the requirements for the degree of

**MASTER OF SCIENCE  
IN  
COMPUTER SCIENCE**

from the

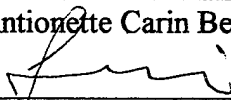
**NAVAL POSTGRADUATE SCHOOL  
March 1997**

Author:



Antionette Carin Bell

Approved by:



Luqi, Thesis Advisor



Valdis Berzins, Co-Advisor



Ted Lewis, Chairman, Department of Computer Science



## **ABSTRACT**

The Computer-Aided Prototyping System (CAPS) is an integrated environment, comprised of an integrated set of software tools, aimed at rapidly prototyping hard real-time embedded systems. The problem with the current CAPS software development environment is the absence of a CAPS User's Manual (How to Use ...), which provides CAPS users with step-by-step guidelines on how to use the CAPS tools.

One solution to this problem was solved by designing, developing, and creating online documentation for the CAPS User Interface and Graphic Editor Reference Manuals for CAPS Release 1.1. Such an approach provides accessible visual, graphical, and textual step-by-step illustrations for CAPS users when interfacing, interacting, and manipulating, commands and options within the CAPS User Interface subsystem and Graphic Editor tool. In addition, this approach includes a glossary which helps CAPS users to understand the meaning of the difficult or specialized terms used in this environment.



## TABLE OF CONTENTS

I. INTRODUCTION .....	1
A.    MOTIVATION .....	1
B.    SCOPE OF THESIS .....	2
C.    ORGANIZATION .....	5
II. CAPS USER INTERFACE REFERENCE MANUAL .....	7
A.    PREFACE .....	8
1.    About This Documentation .....	8
2.    Conventions (Visual Cues) .....	8
3.    Computer-Aided Prototyping System (CAPS) Information .....	10
4.    Audience .....	10
5.    Disclaimer .....	10
6.    How This Manual Is Organized .....	10
B.    CAPS OVERVIEW .....	12
1.    What Is the Computer-Aided Prototyping System (CAPS)? ...	12
2.    Benefits of Using CAPS .....	14
C.    CAPS SOFTWARE TOOLS .....	15
D.    GETTING STARTED .....	17
1.    Starting CAPS for the First Time .....	17
2.    Starting CAPS on a Unix Workstation .....	18



3.	Getting Help .....	19
4.	Exiting CAPS .....	20
E.	USING THE USER INTERFACE .....	21
1.	What Is a User Interface? .....	21
2.	The CAPS User Interface .....	21
3.	Using the CAPS (designer mode) Window .....	22
4.	Understanding the Elements of the CAPS (designer mode) Window .....	23
a.	Window Menu Button .....	24
b.	Icon .....	24
c.	Label .....	24
d.	Menu Bar .....	24
e.	Title Bar .....	24
f.	Work Area .....	24
F.	USING THE CAPS (DESIGNER MODE) WINDOW MENUS .....	25
1.	Why Use Menus? .....	25
2.	Using the CAPS (designer mode) Window Menu Bar .....	25
3.	Using the Help Button .....	26
4.	Using Dialog Boxes .....	27
G.	USING THE CAPS (DESIGNER MODE) WINDOW PULL-DOWN MENUS .....	29
1.	Window Ops Menu .....	29
2.	Prototype Menu .....	31

3.	Edit Menu .....	32
4.	Databases Menu .....	34
5.	Exec Support Menu .....	35
H.	USING THE EDITORS .....	36
1.	What Is an Editor? .....	36
2.	PSDL Editor .....	37
a.	Title Bar .....	38
b.	Window Menu Icons .....	38
c.	Menu Bar .....	38
d.	Divided Windows .....	38
e.	Scroll Bars .....	39
f.	Separator Lines .....	39
3.	Graphic Editor .....	41
4.	Ada Editor and Other Text Editors .....	41
5.	Interface Editor .....	42
6.	Requirements Editor .....	44
7.	Change Request Editor .....	45
I.	USING THE EXECUTION SUPPORT SYSTEM .....	46
1.	Translator .....	46
2.	Scheduler .....	46
3.	Compiler .....	46
J.	USING THE PROJECT CONTROL SYSTEM .....	47

1.	Evolution Control System .....	47
2.	The Merger .....	47
K.	USING THE SOFTWARE BASE .....	48
III.	CAPS GRAPHIC EDITOR REFERENCE MANUAL .....	49
A.	GRAPHIC EDITOR PREFACE .....	49
1.	About This Manual .....	49
2.	What This Reference Manual Does .....	49
3.	Audience .....	50
4.	How This Manual Is Organized .....	50
B.	GETTING STARTED .....	51
1.	Invoking the Graphic Editor .....	51
2.	Executing CAPS .....	52
3.	Invoking the Graphic Editor with a New Prototype .....	54
4.	Invoking the Graphic Editor with an Existing Prototype .....	58
C.	GRAPHIC EDITOR (GE) .....	61
1.	Description .....	61
2.	Usage within CAPS .....	62
3.	User Interface Elements .....	62
4.	Creating a Graph Using the Graphic Editor .....	63
5.	Steps to Create a PSDL Using the Graphic Editor .....	63
6.	Editing a PSDL Graph Using the Graphic Editor .....	64

7.	Saving a Graph within the Graphic Editor .....	65
8.	Printing a Graph within the Graph Editor .....	65
9.	Closing the Graphic Editor .....	66
D.	GRAPH EDITOR INTERFACE .....	67
1.	Description .....	67
2.	Displaying the Graph Editor Interface .....	67
a.	Application Icon or Window Menu Button .....	67
b.	Drawing Space .....	68
c.	Menu Bar .....	68
d.	Scroll Bars .....	68
e.	Status Bar .....	68
f.	Tool Sidebar .....	69
E.	THE APPLICATION ICON AND WINDOW MENU BUTTON .....	71
1.	Description .....	71
2.	Accessing the Application Icon Menu or Window Menu .....	71
3.	User Interface Elements .....	71
F.	THE GRAPHIC EDITOR DRAWING SPACE .....	74
1.	Description .....	74
2.	Usage within CAPS .....	75
3.	User Interface Elements .....	75
a.	Horizontal Scroll Bar .....	75
b.	Down Arrow .....	75

c.	Scroll Box .....	75
d.	Up Arrow .....	76
e.	Vertical Scroll Bar .....	76
f.	Left Arrow .....	76
g.	Scroll Box .....	76
h.	Right Arrow .....	76
G.	THE MENU BAR AND PULL-DOWN MENUS .....	78
1.	Description .....	78
2.	Usage within CAPS .....	78
3.	User Interface Elements .....	78
H.	THE TOOL SIDEBAR .....	84
1.	Description .....	84
2.	Usage within CAPS .....	84
3.	User Interface Elements .....	84
4.	Drawing Tools and Icons .....	85
5.	Properties Tool and Icon .....	87
6.	Stream Tool and Icon .....	88
7.	Select Tool .....	89
IV.	CAPS DEVELOPMENT ENVIRONMENT GLOSSARY .....	91
V.	FUTURE RESEARCH AND DEVELOPMENT .....	115

LIST OF REFERENCES .....	117
INITIAL DISTRIBUTION .....	119

## ACKNOWLEDGEMENTS

First and foremost I would like thank God for allowing this happen; being who He is and surrounding me with people who truly care about my well being as well as my achievements.

I wish to especially thank Professors Luqi and Berzins, my advisors, for all the patience, guidance, and assistance given to complete this thesis. Your support and knowledge is greatly appreciated and has helped me to achieve one of the most difficult challenges in the learning experience.

I wish to thank my newly beloved husband, who so patiently waited for me to complete this thesis giving his love and support during difficult times. Also I wish to thank, Miss Pia S. Boston, for all her encouragement and mentoring during this time.

Last, but not least I wish to thank Mrs. Valerie Brooks, Computer Science System Administrator, and Mrs. Mary Thiele Fobian, Technical Writer, Mrs. Ann Malokas, Software Engineering Research Administrator, and Mickey Harn, PhD student for all their help given to assist me in setting up my CAPS environment, executing CAPS in either the Unix or Microsoft Windows environment in order to retrieve the data needed to complete this CAPS project.

## **I. INTRODUCTION**

### **A. MOTIVATION**

The rapidly growing demand for software has shifted toward larger and more complex systems. An environment to develop software and tools for such systems is the Computer-Aided Prototyping System (CAPS) software development environment. The CAPS environment is an integrated environment comprised of an integrated set of software tools, aimed at rapidly prototyping hard real-time embedded systems [Ref. 4].

The problem with the current CAPS software development environment is the absence of a CAPS User's Manual (How to Use ...), which provides CAPS users with step-by-step guidelines on how to use the CAPS tools. One solution to this problem was solved by designing, developing, and creating online hypertext documentation for the CAPS User Interface and Graphic Editor Reference Manuals for CAPS Release 1.1. Such an approach provides accessible visual, graphical, and textual step-by-step illustrations for CAPS users when interfacing, interacting, and manipulating, commands and options within the CAPS User Interface subsystem and Graphic Editor tool. In addition, this approach includes a glossary which helps CAPS users to understand the meaning of the difficult or specialized terms used in this environment.

The lack of training documentation such as user's guides and reference manuals is evident in missed deadlines. Understanding how to use the software tools in the CAPS environment can help managers manage projects better, can help designers design efficient prototypes, and can help novice CAPS users better understand CAPS and its associated tools.



## **B. SCOPE OF THESIS**

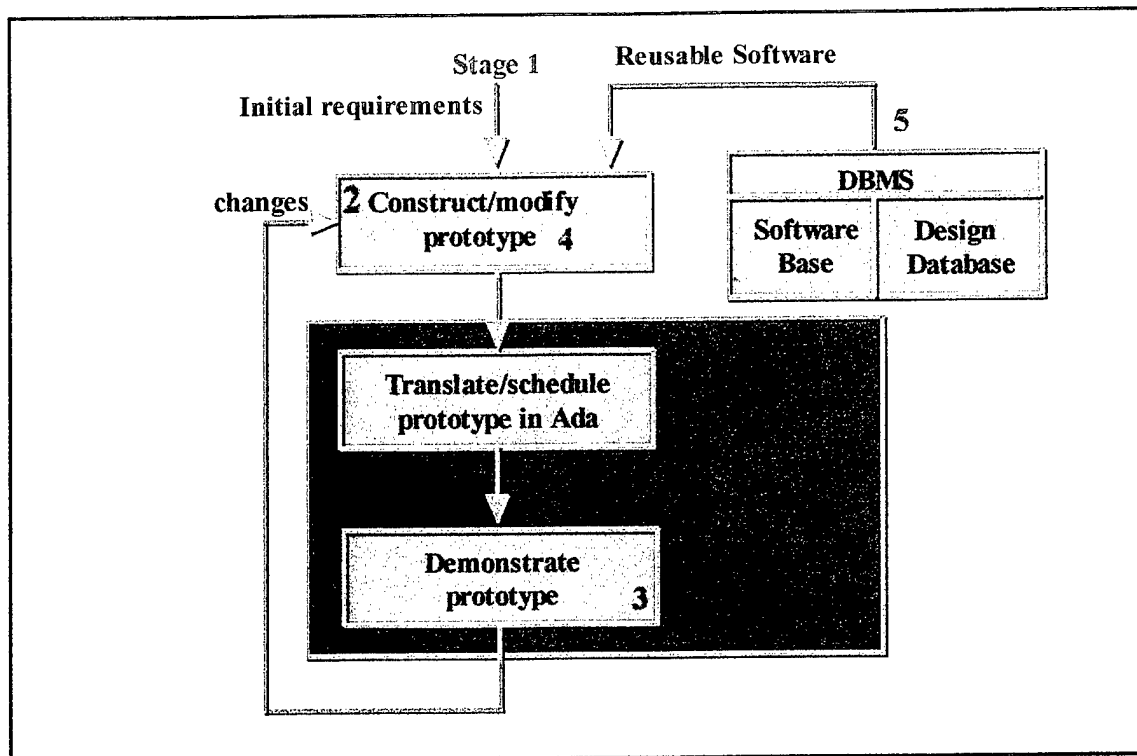
Although [LQ89] illustrates three CAPS subsystems (user interface, software database system, and execution system), this thesis focuses primarily on the user interface subsystem and the graphic editor, one of its tools. This thesis does give a brief overview of the two other subsystems which includes a description and the tools associated with each subsystem. These details are included in the CAPS User Interface Reference Manual.

The goal of this thesis is to assist the CAPS users in understanding CAPS fundamentals, the PSDL language that serves as the foundation of CAPS, and the rapid prototyping approach which was used in the CAPS development environment evolution.

The Prototyping System Description Language (PSDL), which is the basis of CAPS, is used [Ref. 7] in the syntax-directed editor (SDE), the language-based editor, or smart editor, tailored to a specific language, utilizing the grammar, structure, and static semantics of that language to assist the user in writing correct programs [Ref. 11]. PSDL was designed specifically for specifying hard real-time systems. It has a rich set of timing specification features and offers a common baseline from which users and software engineers describe requirements [Ref. 3].

The rapid prototyping approach depends on PSDL and a software base. Rapid Prototyping is an approach to software development that uses prototypes to help both the developers and their customers visualize the proposed system and predict its properties in an iterative process. There are five major stages in the CAPS iterative prototyping process: (1) determine initial requirements for software system design, (2) construction of prototype, (3) execution and demonstration of prototype, (4) adjustments/modifications, debugging, and

iteration of requirements and (5) implementation and optimization of validated requirements.



**Figure 1.1. Iterative prototyping process in CAPS after [Ref. 5].**

As shown in Figure 1.1, initially, the prototype design starts with an analysis of the problem and decision about which parts of the proposed system are to be prototyped. Initial requirements for the prototype are then generated, either informally or in some formal notation. These requirements may be refined by asking users to verify their completeness and correctness. After the requirements analysis is completed, the next step is to use PSDL to specify the prototype according to the requirements analysis. This step involves the construction of data flow diagrams augmented with nonprocedural timing and control constraints and data stream types, resulting in the preliminary, top-level (root) design that is free from programming level details. The underlying computational model unifies dataflow and control flow, and provides a mechanism for developing top-down decompositions. At

this point, the user may continue to decompose any software module until its components can be realized via reusable components drawn from the software base or new atomic components. After decomposition of components, the execution support system, which is outlined and displayed as the shaded area in Figure 1.1, is utilized. The high-level PSDL specification is translated into the target programming language for execution and evaluation to determine if requirements can be validated.

During demonstrations of the prototype, the user evaluates the prototype's actual behavior against its expected behavior. If the prototype fails to execute properly, the user identifies problems and makes changes to redefine the requirements. This process continues until the user determines that the prototype successfully captures the critical aspects of the envisioned system, resulting in what is known as the iterative prototyping cycle [Ref. 1].

Next, during debugging and modification, modifications are assisted by a design database that helps the designers in managing the design history and coordinating the changes. Finally, when the requirements are validated, the designer then uses the validated requirements as a basis for designing the implemented and optimized validated requirements [Ref. 2].

The CAPS User's Interface and Graphic Editor Reference Manual which provides visual, graphical, and textual instructions and illustrations to assist CAPS users when:

- interacting within the CAPS software development environment using a graphical user interface,
- manipulating commands and options using a mouse as the pointing device, and

- interfacing with other CAPS software tools such as the syntax-directed editor and graphic editor.

## **C. ORGANIZATION**

This thesis has five chapters. Chapter II is the User Interface Reference Manual which includes a cover page, a table of contents, and chapter organization that provides step-by-step procedures of using the user interface subsystem. Chapter III is the Graphic Editor Reference Manual which includes also a cover page, a table of contents , and chapter organization that provides step-by-step procedures of how and when to use the tools associated with the Graphic Editor. Chapter IV lists and defines terms and specialize words used in the CAPS development environment. Chapter V discusses recommendations for future CAPS research and training documentation development of its associated tools.



## II. CAPS USER INTERFACE REFERENCE MANUAL

In this thesis, the user interface reference manual basically outlines a simple template format for creating an individual manual. Many resources, for example *OpenWindows Version 3.1 DeskSet Reference Guide*, *How to Use Windows 95*, *WordPerfect 6 for Windows*, *Motif Reference Manual*, etc., were researched and reviewed prior to deciding on the format that was actually used to design, develop, and create this manual.

Basically, this manual introduces and demonstrates X-Windows and Microsoft Windows terminology and user interface elements. Some of the user interface elements in both environments is the same. X-Windows is the windowing environment used to process programs in a Unix environment. Microsoft Windows is the windowing environment used mainly to process programs on a Personal Computer (PC) system. CAPS Release 1.1 is the program that was executed to get the screen captures for the illustrations. This is the current working version of CAPS.

The main purpose of this manual is used to display and illustrate the user interfaces within the CAPS development environment that users encounter and must interact and manipulate to execute the CAPS program which invokes other CAPS tools that are used for creating a new or modifying an existing prototype design or model.

The organization of the sections includes the visual, graphical, and textual contents of the CAPS User's Interface Reference Manual. This reference manual has eleven sections.

## **A. PREFACE**

Before discussing the contents of the user interface manual, a preface section is given. It includes topics such as "About this Documentation," "Conventions (Visual Cues)," "CAPS Information," "Audience," "Disclaimer," and "How This Manual Is Organized."

### **1. About This Documentation**

This *CAPS User's Interface Manual* is a guide to using the user interface subsystem. It includes an overview of CAPS, some of CAPS benefits, and procedures for first-time users. The manual employs consistent visual cues, a few standard text formats, and some user interface and X windows terminology. The term visual cue is used interchangeably with the term convention.

### **2. Conventions (Visual Cues)**

The conventions (visual cues) give the reader at least three indications. That information assist the reader by providing what input style to apply to the procedures used to perform tasks, explaining the meaning of the styles, and indicating what happens when a particular style is used. The following table (Table 2.1) on the following page provides the typographical conventions that are used throughout this manual.

Table 2.1. Convention/Visual Cue Styles and Uses.

Convention/Visual Cue Style	Use
<i>italic</i>	<p>The <i>italic</i> style is used for any data that must be entered exactly as it appears. For example, “<b>type caps</b> at the Unix system prompt” means that the user must type the italicized text exactly as it appears. That does not mean that the text has to be in the italicized font.</p> <p>-or-</p> <p>The <i>italic style</i> indicates the name of publications.</p>
<b>bold</b>	<p>The <b>bold</b> style indicates any action word that requires action from the user. For example, “<b>type caps</b> at the Unix system prompt” means that the word type alerts the user that some action must be performed. Some other action words included in this manual are <b>click</b>, <b>press</b>, <b>select</b>, <b>highlight</b>, etc.</p>
ALL CAPITALS	<p>The ALL CAPITAL style indicates acronyms and names of mouse buttons. Names of mouse buttons are referred to by function, not by location, and are represented in all capital characters. For example, “Press SELECT mouse button” is used versus “Press left mouse button.” This convention is used because the mouse buttons can be customized, so the left mouse button may not be the SELECT button for every user. See the <i>Motif Toolkit User's Guide</i> for instructions on how to customize the mouse button function assignments. See Appendix A for mouse button default settings which are used in this manual.</p>
First/Initial Letter Capitals	<p>This style indicates the single or compound name or labels of buttons, menu names, and keyboard keys. For example, Save is a single name of a button; File is the single name of menu; Enter is the single name of a keyboard key and Print Screen is an example of a compound name of a keyboard key.</p>



### **3. Computer-Aided Prototyping System (CAPS) Information**

CAPS was developed by Professor Luqi and the CAPS group at the Naval Postgraduate School. To obtain answers to your questions or additional information, or to send comments about CAPS Release 1.1, you can send e-mail to: [caps@cs.nps.navy.mil](mailto:caps@cs.nps.navy.mil).

### **4. Audience**

This manual is designed to assist all managers and developers. The novice as well as advanced users can use this manual for assistance while quickly developing, analyzing, and refining requirements for real-time and embedded systems.

### **5. Disclaimer**

CAPS is an ongoing research project that consists of many individual research efforts. Consequently, CAPS is a dynamic and diverse system. Many CAPS components have been developed in near isolation and incorporated onto the system as a whole. Every reasonable effort has been made to make CAPS robust and user-friendly; however, due to the very nature of its development environment, it is not perfect. Many aspects of CAPS could be ergonomically, semantically, procedurally or otherwise improved. The CAPS development team is well aware of this. Our intent has been not to create an industrial strength product, but rather to put to the test (and to the silicon) new ideas, concepts and philosophies from an academic environment.

### **6. How This Manual Is Organized**

This *CAPS User's Interface Reference Manual* describes CAPS capabilities and steps required to use its software tools and its user interface elements. The information is organized into eleven first order sections for this thesis.

- **SECTION A-PREFACE**-Contains the introduction of the CAPS User's Interface Reference Manual.
- **SECTION B - CAPS OVERVIEW** - Gives an overview of CAPS and lists CAPS benefits.
- **SECTION C-CAPS SOFTWARE TOOLS**- Introduces the set of integrated tools used within the CAPS software development environment.
- **SECTION D-GETTING STARTED** - Lists the steps necessary to begin using CAPS for the first time or on a Unix workstation. Also gives help and exiting CAPS tips.
- **SECTION E-USING THE USER INTERFACE** - Defines the term user interface and familiarizes the user with the CAPS user interface subsystem and the user interface elements associated with it.
- **SECTION F-USING THE CAPS (DESIGNER MODE) WINDOW MENUS**- States why menus are used and illustrates the manipulation of various user interface elements such as active windows, icons, labels, dialog boxes, pull-down menus, etc. to perform interactive tasks. In addition, this Section provides instructions on how to use these features to assist users that are using this manual and creating or modifying prototype designs using the CAPS software tools.
- **SECTION G-USING THE CAPS (DESIGNER MODE) PULL-DOWN MENUS** - Describes the pull-down menus accessible from the CAPS (designer mode) interface. Displays the functions of the pull-down menu button in a table.
- **SECTION H-USING THE EDITORS**-Identifies the different editors provided within the CAPS development environment.
- **SECTION I-USING THE EXECUTION SUPPORT SYSTEM**-Defines the execution support system and the usage of this subsystem's elements within the CAPS development environment.

- **SECTION J-USING THE PROJECT CONTROL SYSTEM**-Explains the use of the project control system. States its components and their usage in the CAPS development environment.
- **SECTION K-USING THE SOFTWARE BASE**-Addresses its existence as a one of CAPS subsystems and refers to different sources that gives a definition of the software base.

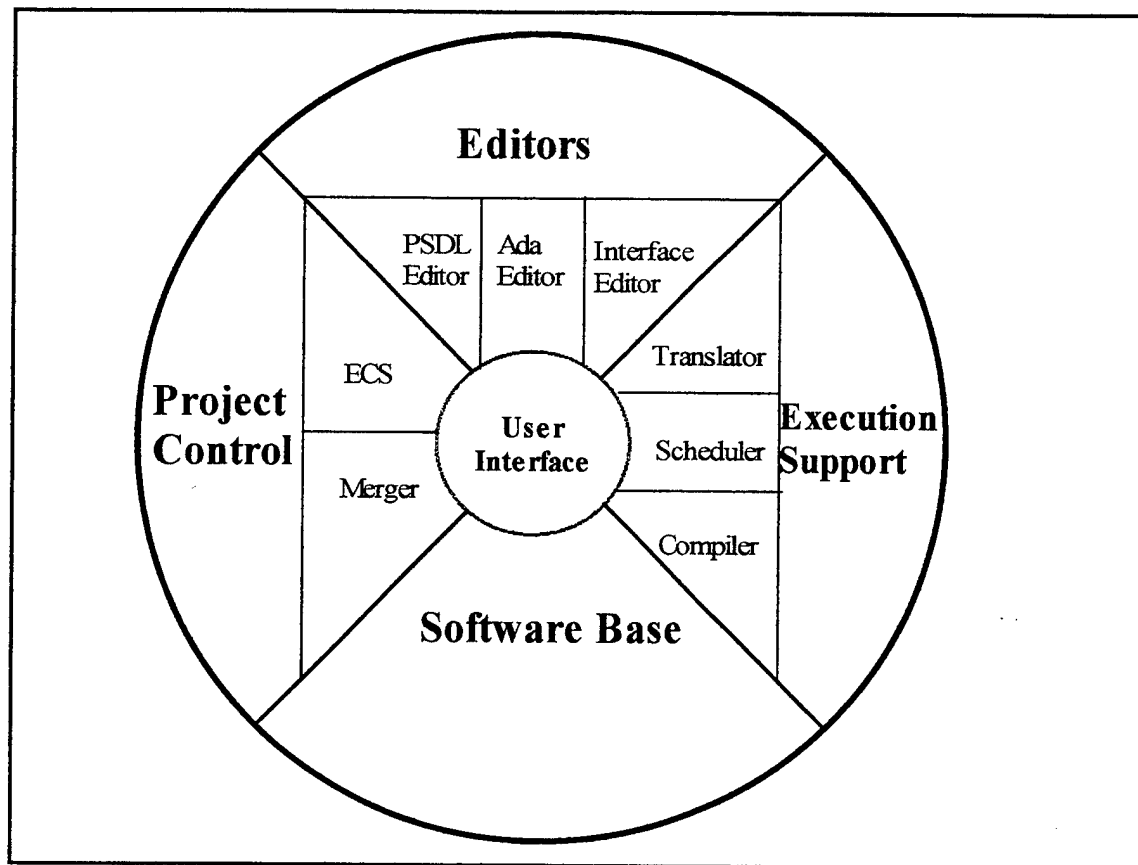
The format used above was taken from the Dennis Michael Trepanier [DMT95] master thesis *Internetworking: Recommendations on Network Management for K-12 schools*.

## **B. CAPS OVERVIEW**

### **1. What Is the Computer-Aided Prototyping System (CAPS)?**

The Computer-Aided Prototyping System (CAPS) is a set of integrated software engineering tools for creating real-time systems prototypes. Such prototypes are useful for requirements analysis, feasibility studies and systematic development of embedded systems.

CAPS provides managers and designers with the capability tool to quickly develop, analyze, and refine real-time software systems. Its general structure is shown in Figure 2.1.



**Figure 2.1. CAPS development environment general structure after [Ref. 11].**

The CAPS Release 1.1 program primarily relies on the user-friendliness of three interfaces the “CAPS (designer mode)” window, the “psdl\_editor” window, and the Graph Editor window for communication and interaction between CAPS, the users/software engineers and the CAPS software tools. The first window is discussed and illustrated in Section E of this manual. The second window is discussed in Scott Grosenheider [SG95] master thesis. The third window is discussed and illustrated in the Graphic Editor Reference Manual.

## **2. Benefits of Using CAPS**

The primary objective of the CAPS system is to assist Department of Defense (DOD) program managers and engineers to rapidly evaluate requirements for real-time control software using executable prototypes. Other objectives include testing and integrating completed subsystems through evolutionary prototyping.

CAPS provides a capability to quickly develop functional prototypes so that feasibility can be verified early in the software development process; however, its most significant role is automatically generating code to increase productivity at a very low cost.

Other functionality includes:

- Supporting integration of large complex systems from requirements to maintenance
- Formulating /Validating requirements via prototype demonstration and user feedback
- Assessing feasibility of real-time system designs

The benefits of using evolutionary prototyping supported by CAPS include:

- Reduced risks of project failure
- Validated systems that meet the stakeholders needs
- Lowered reliability and maintenance costs
- Reduced system integration problems
- Improved responsiveness to changing requirements

## **C. CAPS SOFTWARE TOOLS**

The CAPS software tools are divided into 5 categories:

1. User Interface
2. Editors
3. Execution Support System
4. Project Control System
5. Software base

Each of the tools are part of a CAPS subsystem. Three of CAPS main subsystems are shown in Figure 2.2. The subsystems are the user interface, the software database, and the execution support system. At the lower levels each of the subsystems are shown with its associated software tools. The shaded box depicts the items that are discussed in this manual.

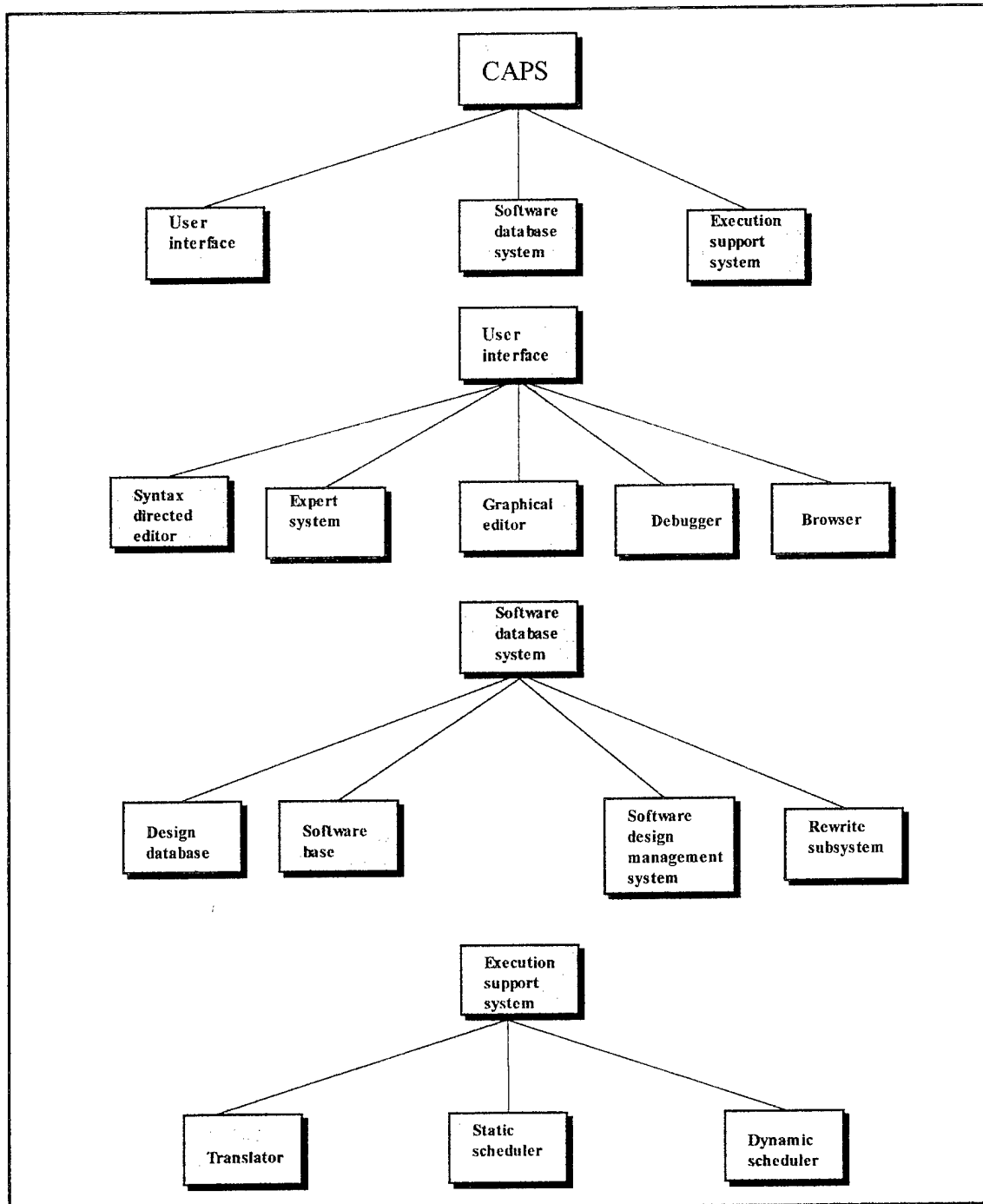
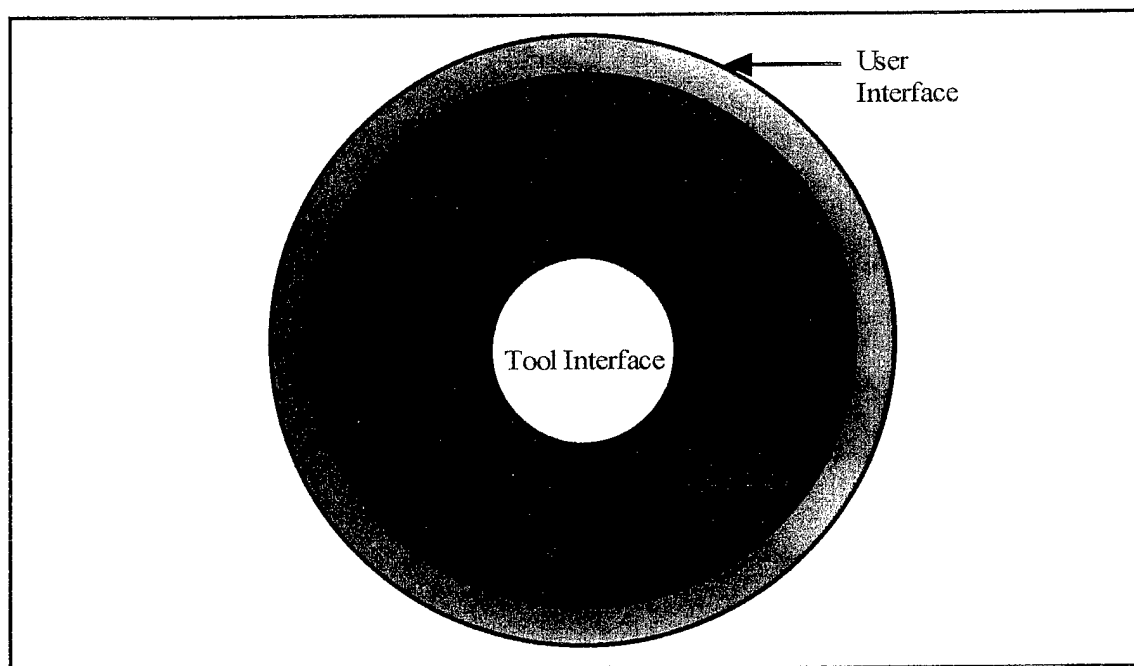


Figure 2.2. CAPS software tools.

These tools are further categorized into either essential, very useful, or useful tools. Compilers, operating systems, assemblers, and programming languages are essential tools.

Computer system utilities editors, libraries, and the library of reused code are very useful tools. Finally, computer-aided software engineering (CASE) tools are among the group of useful software development tools.

Figure 2.3 illustrates the CAPS advanced rapid prototyping environment as discussed in [LS94]. This environment is supported by previously mentioned software tools. Also, it the CAPS approach used to identify problems early in the development process.



**Figure 2.3. CAPS advanced rapid prototyping environment after [Ref.6].**

#### **D. GETTING STARTED**

##### **1. Starting CAPS for the First Time**

Obtain an account (login id and password) from your System Administrator. Use your login id and password to access a Unix workstation.

After successfully accessing a Unix workstation, check your *.cshrc* file to ensure your environment is set up properly to execute CAPS. The instructions to set up your caps



environment are shown in Figure 2.4.

```
# set up for caps

setenv CAPSHOME /local/caps

if(-d $CAPSHOME) then

    set path = ($path $CAPSHOME/bin)

    source $CAPSHOME/bin/CAPSsetup

endif
```

**Figure 2.4. Instructions to set up the CAPS environment.**

If your `.cshrc` file contains the instructions shown in Figure 2.4, you are ready to start CAPS. Otherwise, insert the instructions to set up your CAPS environment in your `.cshrc` file. You may have to substitute the path name for the location of CAPS in your installation for “local/caps.”

After editing the `.cshrc` file, save it using the Save command within the active editor you are executing.

Close the active editor or activate an “xterm” window by placing the mouse pointer (arrow) directly over it and at the host Unix system prompt (for example, `se7>`), an active cursor appears in the window. Type `source .cshrc` at the Unix system prompt (for example, `se7>source .cshrc`). Sourcing the file makes your modifications take effect.

## **2. Starting CAPS on a Unix Workstation**

After successfully logging in and setting up the CAPS software development environment start CAPS by typing `caps` at the Unix system prompt. The “CAPS (designer

mode)” window as shown in Figure 2.5 appears on the screen.

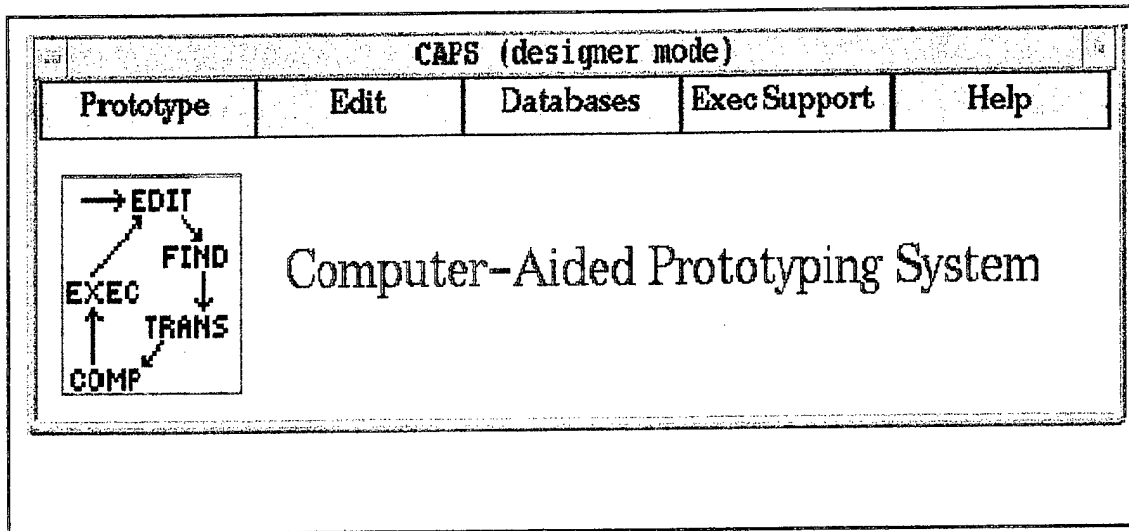


Figure 2.5. CAPS (designer mode) window.

**NOTE:** The “CAPS (designer mode)” window remains open while the system is being loaded in a window known as an “xterm” execution window. While CAPS is being loaded, a process is running in the “xterm” execution window. This is called a background process. The background process causes the “xterm” execution window to display a record of system actions on the screen.

The “CAPS (designer mode)” window provides the user with an interface to access other CAPS software tools via five pull-down menus. Each menu is discussed later.

### 3. Getting Help

Help is provided via previous CAPS users, *The CAPS Quickstart* and *The CAPS Tutorial*, online support, and the Help button.

The Help button is the last button located on the menu bar. This user interface element is discussed and illustrated in Section E.

Perhaps the best help can be classified as that which is provided by the previous CAPS

users, they can point out lesson learned.

#### 4. Exiting CAPS

To exit CAPS perform the steps below:

**Step 1:** Close all other open CAPS windows first.

**Step 2:** From the “CAPS (designer mode)” window, select the Prototype menu item from the menu bar. Its pull-down menu with its default item highlighted appears on the screen just below it. Move the mouse cursor down the pull-down menu highlighting each button until desired choice is located. Figure 2.6 displays Prototype pull-down menu with the Quit button highlighted.

**NOTE:** The window below was executed on a PC and the location of the window title is different. This window displays an application icon versus the control-menu box and a minimize, maximize, and close button are located near the right border of the window.

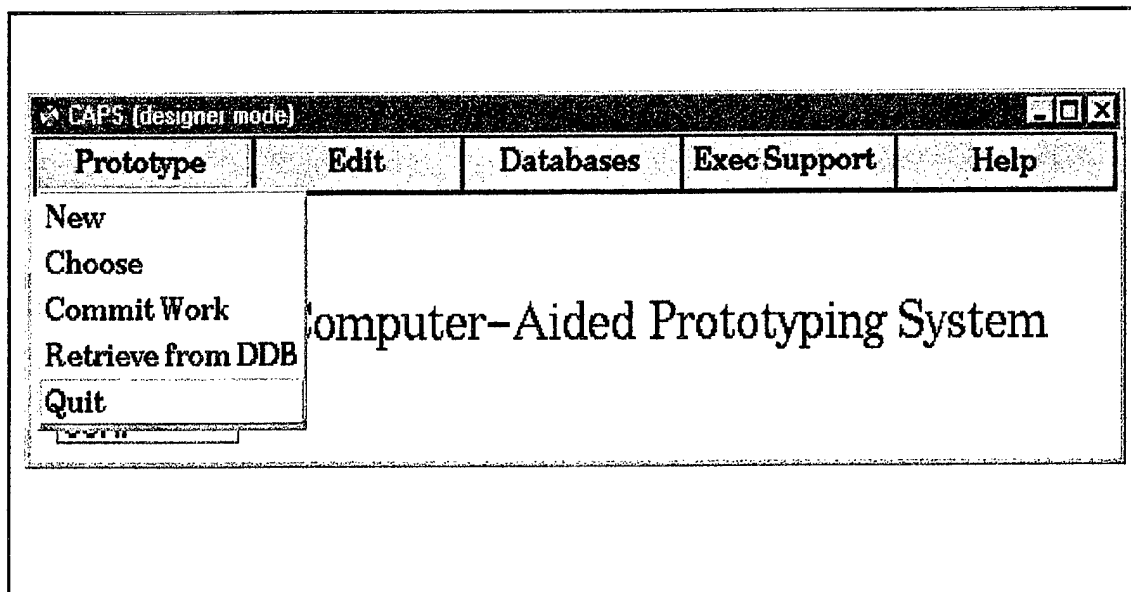


Figure 2.6. Prototype pull-down menu with Quit button highlighted.

**Step 3:** Select highlighted Quit button from the Prototype pull-down menu.

**Step 4:** Watch the “xterm” window, it displays “Quitting CAPS” message.

**Step 5: Activate** the “xterm” window by positioning the mouse cursor within its boundaries and press the Enter or Return key on the keyboard to accept the message. The Unix system prompt will appear on the screen.

## **E. USING THE USER INTERFACE**

### **1. What Is a User Interface?**

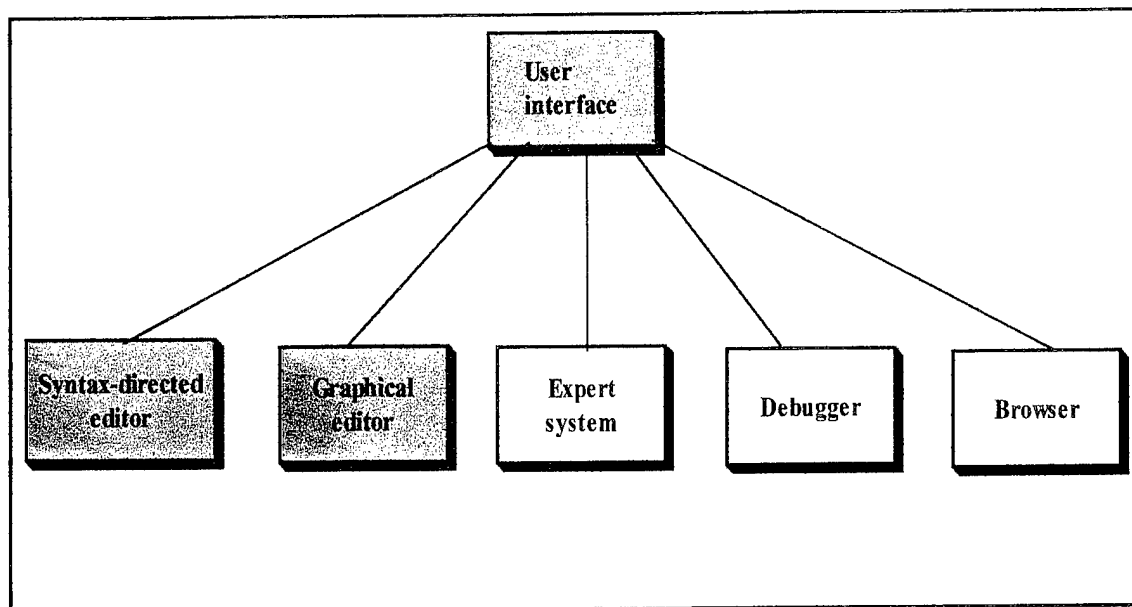
A user interface is that part of a program with which users interact. It consists of all means by which a program communicates with the user, including command line, menus, dialog boxes, windows, lists, and so on. In CAPS, because graphical elements such as buttons, icons, and scroll bars are used with a mouse, this tool is said to have a graphical user interface.

The graphical user interface allows users to select programs, files, or commands by pointing to pictorial representations on the screen rather than typing, long complex commands from a command prompt.

Subsystem programs execute in windows, using a consistent set of pull-down menus, dialog boxes, and other graphical elements as mentioned previously.

### **2. The CAPS User Interface**

The user interface in CAPS is one of its subsystems. It consist of tools such as a syntax-directed editor, a graphical editor, an expert system, a debugger, and a browser. These tools are illustrated in Figure 2.7. The shaded blocks depicts the CAPS subsystem and its associated tools that are discussed in this manual. The graphic editor and syntax-directed editor together provide a user-friendly environment for the user/software engineer to construct a prototype using a combination of graphical and textual objects [LS94].



**Figure 2.7. CAPS User Interface software tools.**

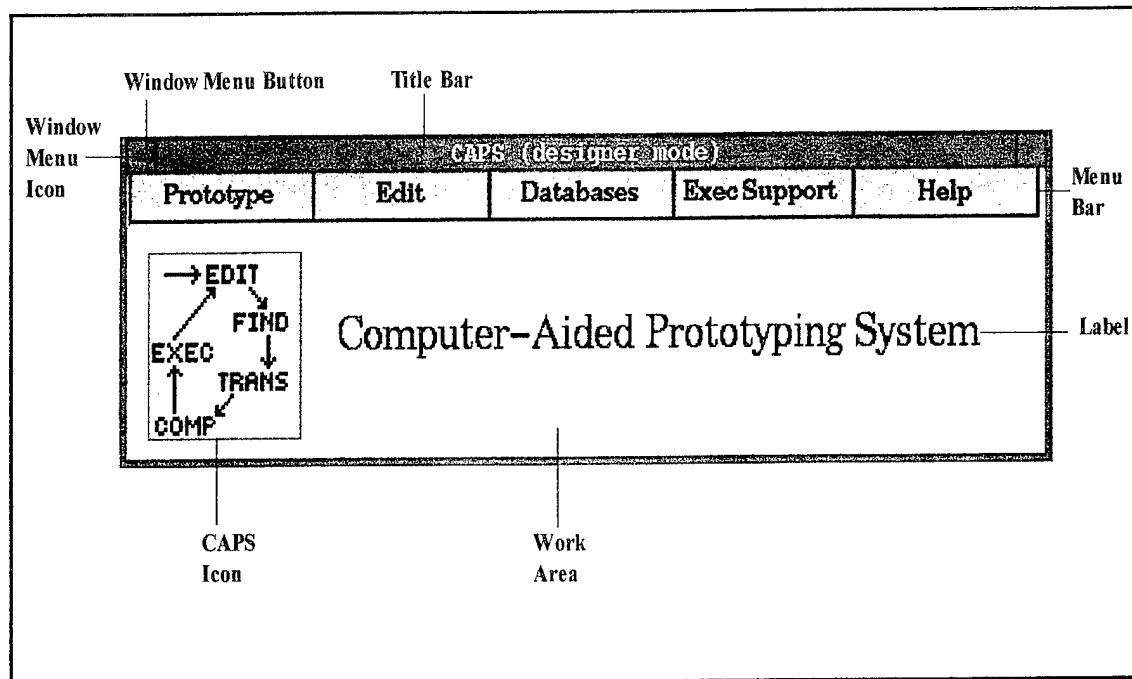
### **3. Using the CAPS (designer mode) Window**

Before you begin creating or modifying any prototype, you must first perform the steps for starting CAPS for the first time or starting CAPS from a Unix workstation.

**NOTE:** The computers in the CAPS software development environment are Unix workstations.

The “CAPS (designer mode)” window is the first window that appears on the screen after CAPS has been executed. Figure 2.8 shows this window with its interface elements labeled with a sequence of characters identifying it.

This is the window that allows and controls communication between CAPS, Project Managers, Technical Developers and CAPS software tools used in this software development environment.



**Figure 2.8. Labeled CAPS (designer mode) window.**

#### **4. Understanding the Elements of the CAPS (designer mode) Window**

When you start CAPS in a Unix environment, your computer will display Figure 2.8 without the labels. You will need to understand the functionality of these elements in order to interface and interact with other CAPS software tools to create and edit prototypes. It is a good idea for users to familiarize themselves with X Windows and basic standard Microsoft Windows terminology.

The "CAPS (designer mode)" interface, shown above has graphical user interface elements such as a window menu button, icons (Window Menu icon and CAPS icon), labels (five menu names and system name), a menu bar (five menu items), a title bar, and a work area. See CAPS Development Environment Glossary for meaning of terms.

**a.      *Window Menu Button***

In a Unix environment, the Window Menu button is a window manager element that contains menu commands for closing, moving, sizing, maximizing, and minimizing.

Associated with the Window Menu button is a window menu (pull-down menu). The window menu is the menu that appears when the user click on the Window Menu button at the top left corner of any X window. The window menu displays a list of operations that are used to change the overall size and shape of the active window.

**b.      *Icon***

An icon displays a small picture or graphic that represents a program, a command, or a piece of information.

**c.      *Label***

A label displays a sequence of characters which displays the name of a system, the name of menu items, etc. For example, "Computer-Aided Prototyping System."

**d.      *Menu Bar***

The menu bar displays menu names

**e.      *Title Bar***

The title bar displays a sequence of text showing the name of the active window.

**f.      *Work Area***

The work area is the user interface element of the window where the user places the graphical user interface elements used for communication and interaction between

the user and other interface elements.

## **F. USING THE CAPS (DESIGNER MODE) WINDOW MENUS**

### **1. Why Use Menus?**

The "CAPS (designer mode)" window uses what is called pull-down menus. The pull-down menus are used to give commands that tells CAPS which tool is executing and what task the tool is performing. In addition, the pull-down menus offer lists of menu commands to choose from. As previously illustrated in Figure 2.8, the menu names are located on the menu bar near the top of the window. When you select a menu name, a menu pulls-down or opens up to reveal a list of menu commands.

You can use the mouse to select menu commands. **Notice** that each menu name is selected by positioning the mouse pointer on the menu command button of choice. The menu command button is highlighted with a highlight bar to indicate that the menu command is currently selected. In the "CAPS (designer mode)" interface, the highlight bar is a hollow rectangular box. Menu commands displayed as dimmed text are not available for selection at the present time.

### **2. Using the CAPS (designer mode) Window Menu Bar**

The "CAPS (designer mode)" menu bar displays the names of five menu items (buttons). The buttons include the Prototype button, an Edit button, a Databases button, an Exec (short for Execution) Support button, and a Help button.

In addition, the "CAPS (designer mode)" pull-down menus utilizes the menu bar buttons shown in Figure 2.8 to access the other CAPS software tools. The pull-down menus are discussed and illustrated later in Section G.



### 3. Using the Help Button

The Help button is the last button on the menu bar. When this button is selected no menu appears on the screen. Moreover, the button is displayed in reverse video (Figure 2.9) and the mouse pointer changes from an arrow to a question mark icon. Reverse video means that the foreground and background is displayed in reverse of each other. The reverse video highlights the help button and indicates that it was selected.

After selecting the help button, the floating question mark is used as the mouse pointer. The question mark icon indicates that the CAPS program is in a help mode. The movement of the mouse allows the question mark icon to float around on the screen, selecting menu names from the menu bar to get help.

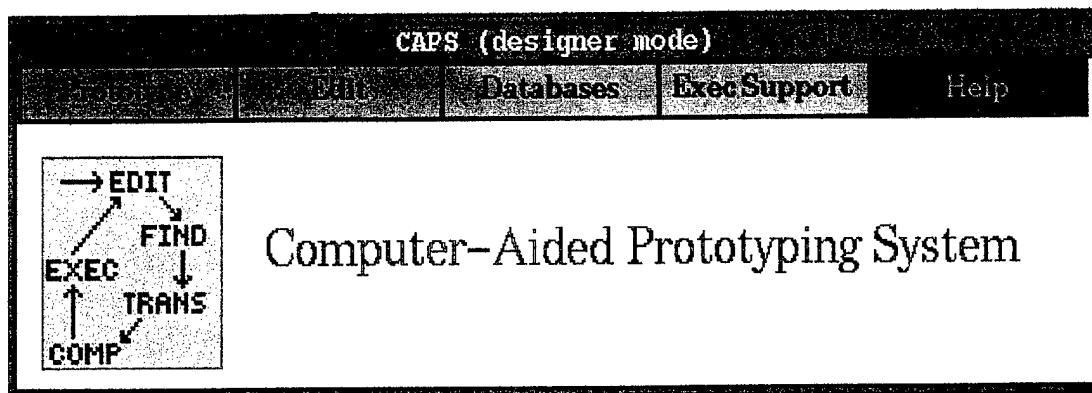


Figure 2.9. Help button highlighted.

To get help on one of the menu items, simply, **position** the mouse pointer on any of the menu names in the "CAPS (designer mode)" window and **click** the SELECT mouse button. A dialog box similar to Figure 2.10 appears on the screen presenting information about the menu item that was selected.

Although the "CAPS (designer mode)" Help is not an extensive help system, it does

provide information about each menu command button on each pull-down menu. Other interface element displayed associated with the dialog box include scroll bars, and a carat symbol. The text box and Close button are dialog box elements. Use the scroll bars to scan the information. Use the carat symbol as an insertion pointer for the text. Use the text box to enter desired text. Use the Close button to close the "CAPS (designer mode)" Help.

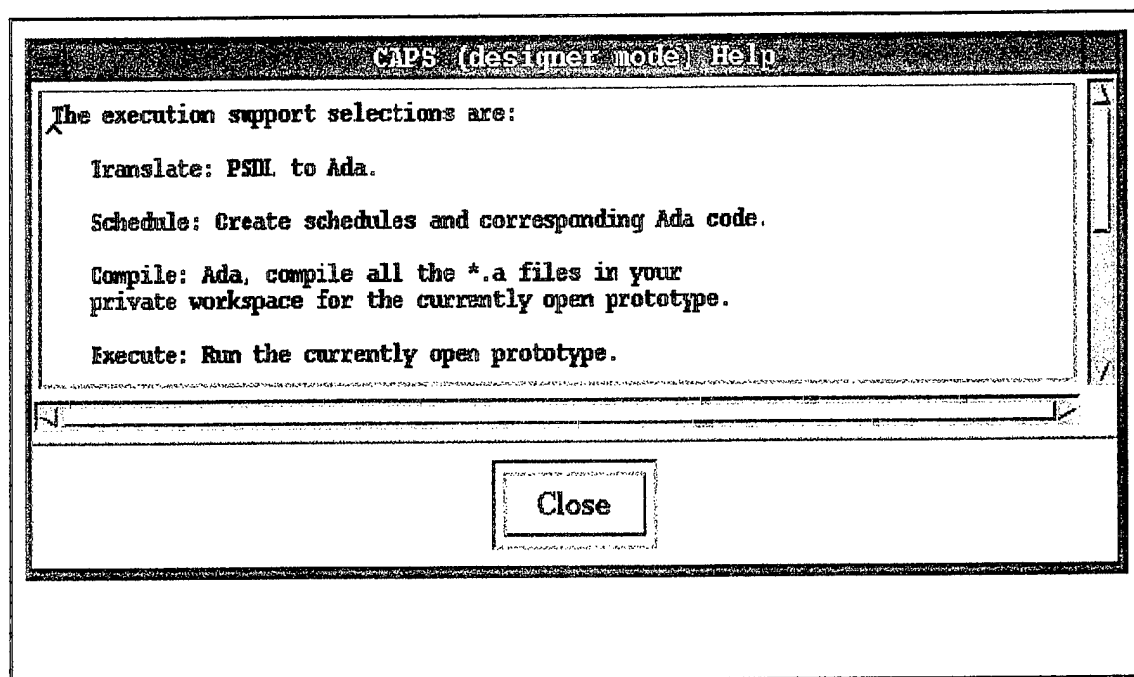


Figure 2.10. Execution Support help information.

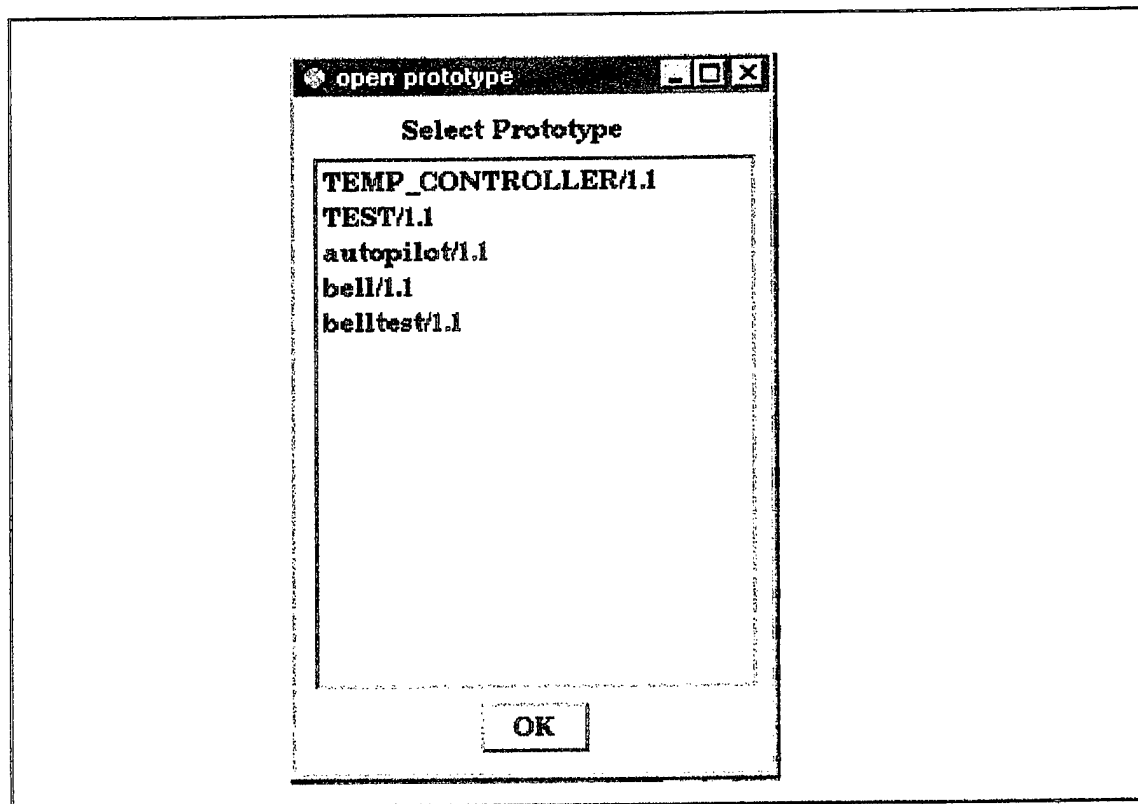
#### 4. Using Dialog Boxes

A dialog box is a separate window that appears when a CAPS program needs additional information to carry out a command. Both menu commands and command buttons opens a dialog box. Each dialog box is different, but they all share common elements.

Figure 2.11 shows the Open Prototype dialog box. The dialog box elements include a list box, and a command button.

**NOTE:** Figures 2.11 and Figure 2.12 both display an application icon in the left corner of the window and a minimize button, a maximize button, and a close button near the right corner of the window. These elements indicate that CAPS was executed in a standard Windows environment.

The command button (Close) is used to execute the dialog box. To select this button, position the mouse pointer on the button and **click** the SELECT mouse button or press the Enter/Return key on the keyboard. When this button is clicked the window disappears from the screen.



**Figure 2.11. Open prototype dialog box.**

The list box is used to select one item in the list by highlighting it. To select with the mouse, simply, **click** the desired item from the list. Use the scroll bar to display all the items in the list if needed.

Figure 2.12 shows the New Prototype dialog box. The dialog box elements include a text box, and two command buttons (Ok and Cancel).

The text box is used to enter information, **click** the box and **type** the appropriate text. Use the Delete (Del) key or the Backspace key to edit your entry.

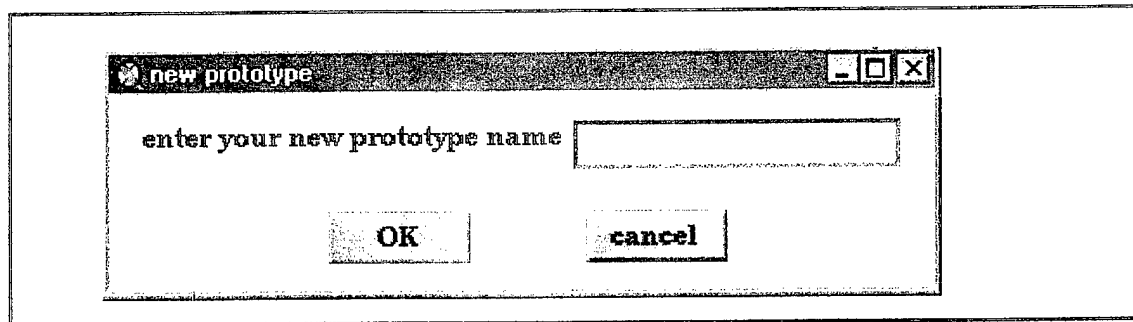


Figure 2.12. New prototype dialog box.

## G. USING THE CAPS (DESIGNER MODE) WINDOW PULL-DOWN MENUS

### 1. Window Ops Menu

The Window Ops (short for Operations) menu shown in Figure 2.13 offers a list of menu commands to choose from which include Close, Refresh Window, Move, Resize Window, Raise Window, Lower Window, Raise or Lower, Maximize, Normal, and Kill Window. The table (Table 2.2) that follows lists each menu command button and its function.

**NOTE:** The Normal button appears dimmed to indicate that this option is not available at the present time. All other options are available and applied to the active window. To select the Window Ops menu and select a menu command from the displayed menu, follow the

simple steps below:

**Step 1: Position** the mouse pointer over the window menu icon. This is the button location in the left corner of the window. The Window Ops menu pulls down on the screen just below the window menu button.

**Step 2: Highlight** your desired menu command.

**Step 3: Click** the SELECT mouse button to select the menu command chosen.

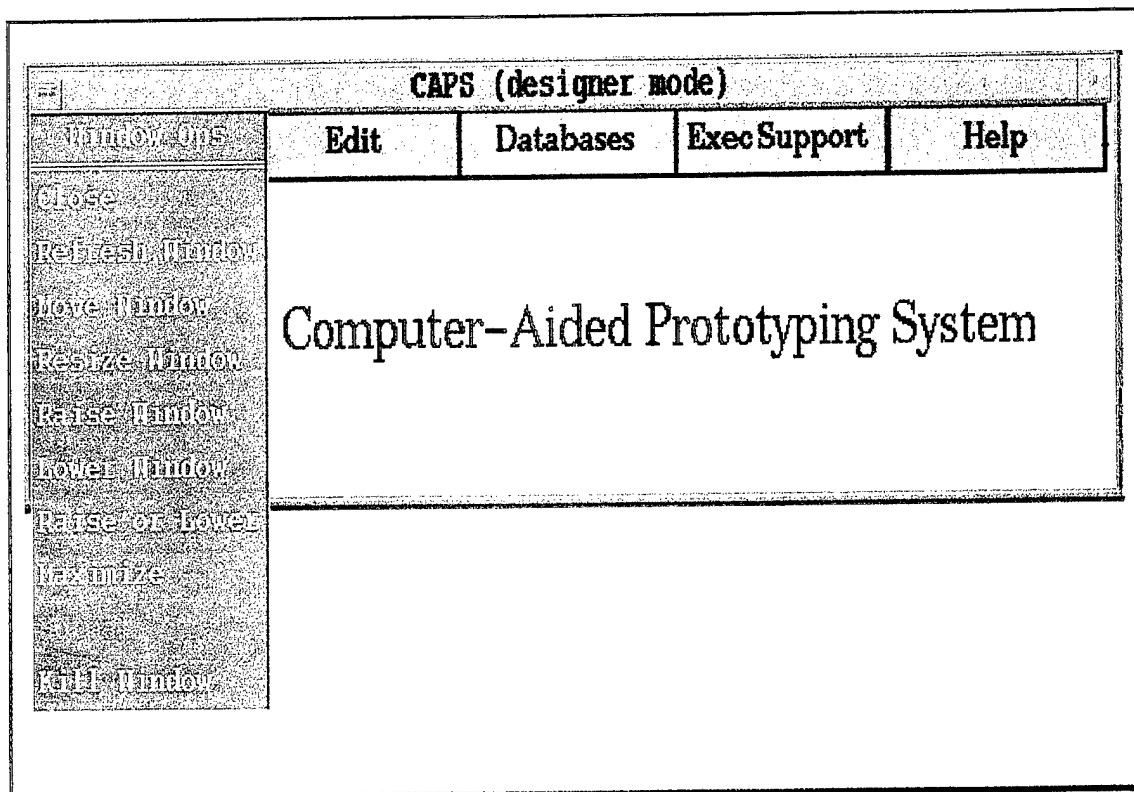



Figure 2.13. Window Operations pull-down menu.

**Table 2.2. Window Ops Menu Command Buttons and Functions.**

<b>Menu Command Button</b>	<b>Function</b>
	<i>Closes the active window or screen.</i>
	<i>Causes the active window or screen to disappear from the screen and returns to the screen in a few seconds.</i>
	<i>Moves the active window to another location on the screen.</i>
	<i>Changes the size of the active window.</i>
	<i>Puts the active window to the foreground of the screen.</i>
	<i>Sends the active window to the background of the screen.</i>
	<i>Puts or sends the active window to the foreground or background of the screen.</i>
	<i>Enlarges the size of the active window.</i>
	<i>Dimmed means option not available at present time. Returns window to original size.</i>
	<i>Exits the active window.</i>

## **2. Prototype Menu**

The Prototype menu shown in Figure 2.14 offers a list of menu commands to choose from which include New, Choose, Commit Work, Retrieve from DDB and Quit. The table (Table 2.3) that follows lists each menu command button and its function. To select the Prototype menu and select a menu command from the displayed menu, follow the simple steps below:

**Step 1: Position** the mouse pointer over the Prototype button on the menu bar. A hollow rectangular outline surrounds the menu name and the default menu choice.

**Step 2: Highlight** your desired menu command.

Step 3: Click the SELECT mouse button to select the menu command chosen.

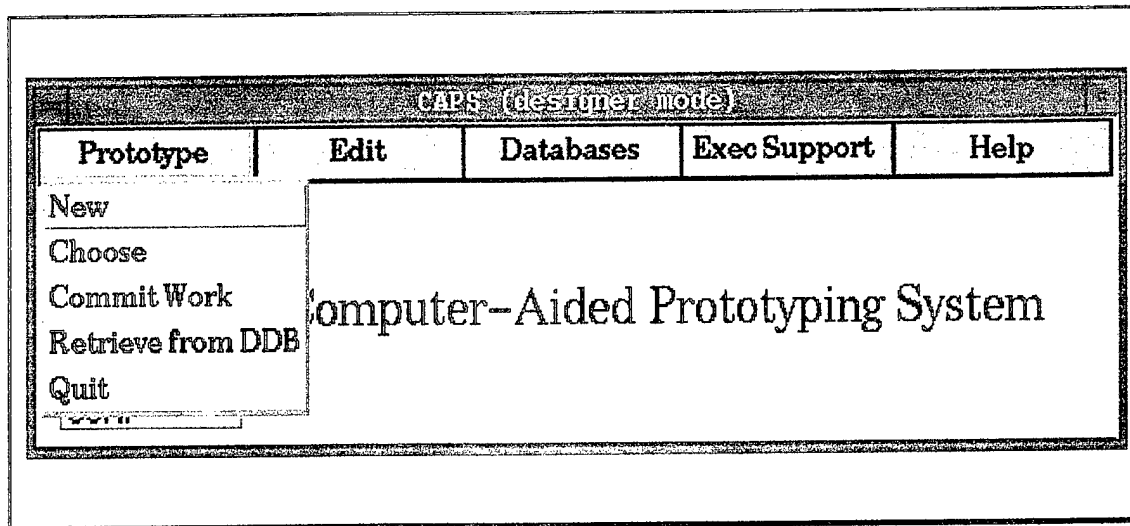


Figure 2.14. Prototype pull-down menu.

Table 2.3. Prototype Menu Command Buttons and Functions.

Menu Command Button	Function
<i>New</i>	<i>Allows the user to create a prototype design.</i>
<i>Choose</i>	<i>Allows the user to modify an existing prototype design.</i>
<i>Commit Work</i>	<i>Allows prototype design to be entered into the database. Not implemented in CAPS Release 1.</i>
<i>Retrieve from DDB</i>	<i>Allows user to retrieve data from the database. Not implemented in CAPS Release 1..</i>
<i>Quit</i>	<i>Quits and closes the CAPS program.</i>

### 3. Edit Menu

The Edit menu shown in Figure 2.15 offers a list of menu commands to choose from which include PSDL, Ada, Interface, Requirements, Change Request, CAPS Defaults, and Hardware Model. The table (Table 2.4) that follows lists each command button and its function.

To select the Edit menu and select a menu command from the displayed menu, follow the simple steps below:

**Step 1: Position** the mouse pointer over the Edit button on the menu bar. A hollow rectangular outline surrounds the menu name and the default menu choice.

**Step 2: Highlight** your desired menu command.

**Step 3: Click** the SELECT mouse button to select the menu command chosen.

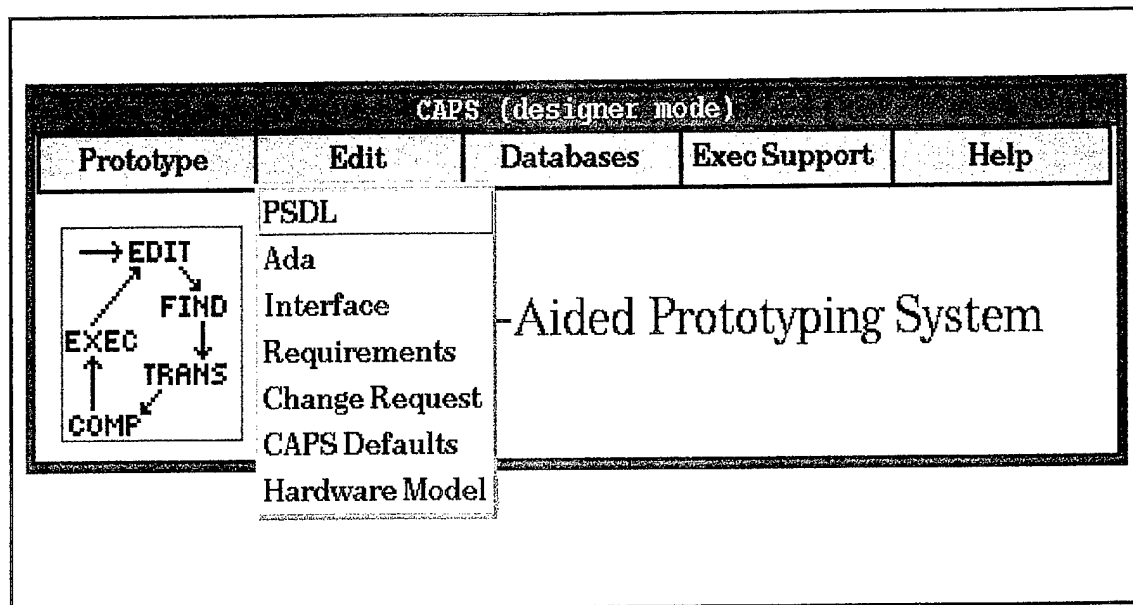
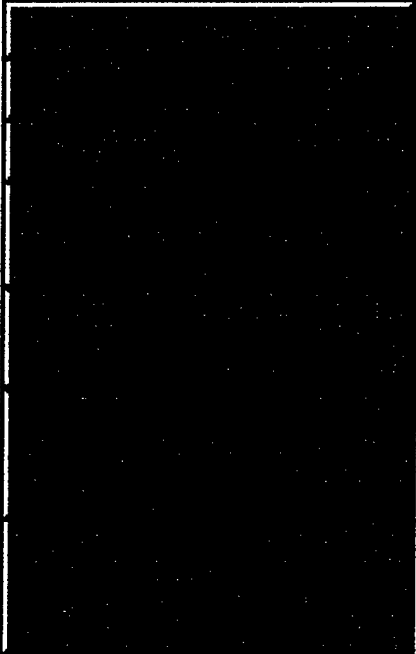


Figure 2.15. Edit pull-down menu.



**Table 2.4. Edit Menu Command Buttons and Functions.**

Menu Command Button	Function
	<i>Invokes CAPS Graphic Editor.</i>
	<i>Allows designers to edit Ada implementation files.</i>
	<i>Invokes TAE+ to edit the prototype interface.</i>
	<i>Allows designers to edit requirements file using the editor specified in the CAPS Defaults window.</i>
	<i>Allows designers to edit a request file using the editor specified in the CAPS Defaults window.</i>
	<i>Allows designers to choose which text or Ada editor will be used; gives designers the option to view or not to view scheduler diagnostics.</i>
	<i>Lets designers check timing constraints relative to a machine faster or slower than the machine which is executing CAPS.</i>

#### **4. Databases Menu**

The Databases menu shown in Figure 2.16 offers a list of menu commands to choose from which include Design Database and Software Base. These tools are not yet implemented in CAPS release 1. The table (Table 2.5) that follows lists each menu command button and its implementation information as its function to inform CAPS users.

To select the Databases menu and select a menu command from the displayed menu, follow the simple steps below:

**Step 1: Position** the mouse pointer over the Databases button on the menu bar. A hollow rectangular outline surrounds the menu name and the default menu choice.

**Step 2: Highlight** your desired menu command.

**Step 3: Click** the SELECT mouse button to select the menu command chosen.

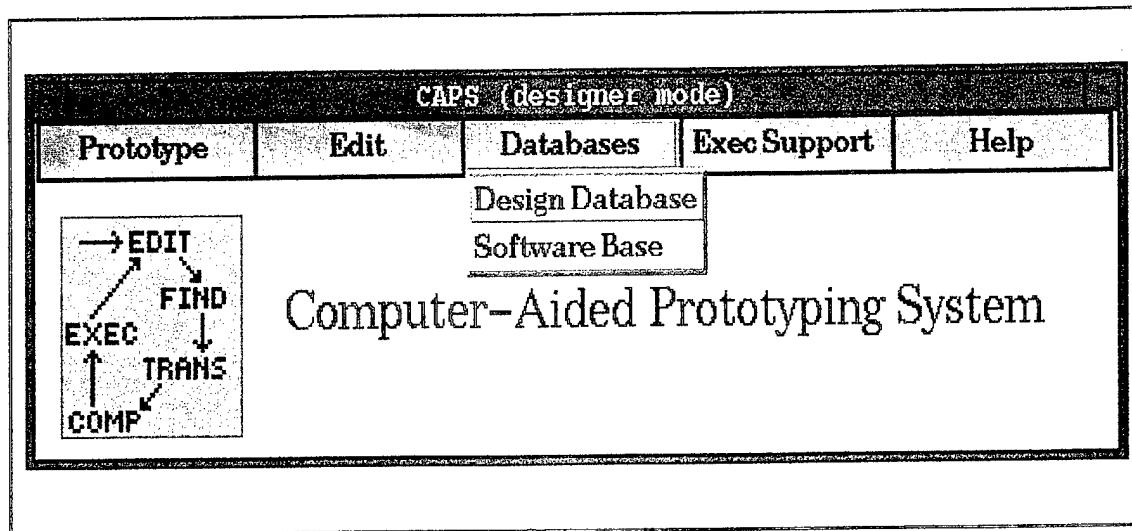


Figure 2.16. Databases pull-down menu.

Table 2.5. Databases Menu Command Buttons and Functions.

Menu Command Button	Function
<i>Designer Database</i>	<i>Not implemented in CAPS release 1.</i>
<i>Software Base</i>	<i>Not implemented in CAPS release 1.</i>

## 5. Exec Support Menu

The Exec (short for Execution) Support menu shown in Figure 2.17 offers a list of menu commands to choose from which include Translate, Schedule, Compile, and Execute. The table (Table 2.6) that follows lists each command button and its function.

To select the Exec Support menu and select a menu command from the displayed menu, follow the simple steps below:

**Step 1:** Position the mouse pointer over the Exec Support button on the menu bar. A hollow rectangular outline surrounds the menu name and the default menu choice.

**Step 2:** Highlight your desired menu command.

**Step 3:** Click the SELECT mouse button to select the menu command chosen.

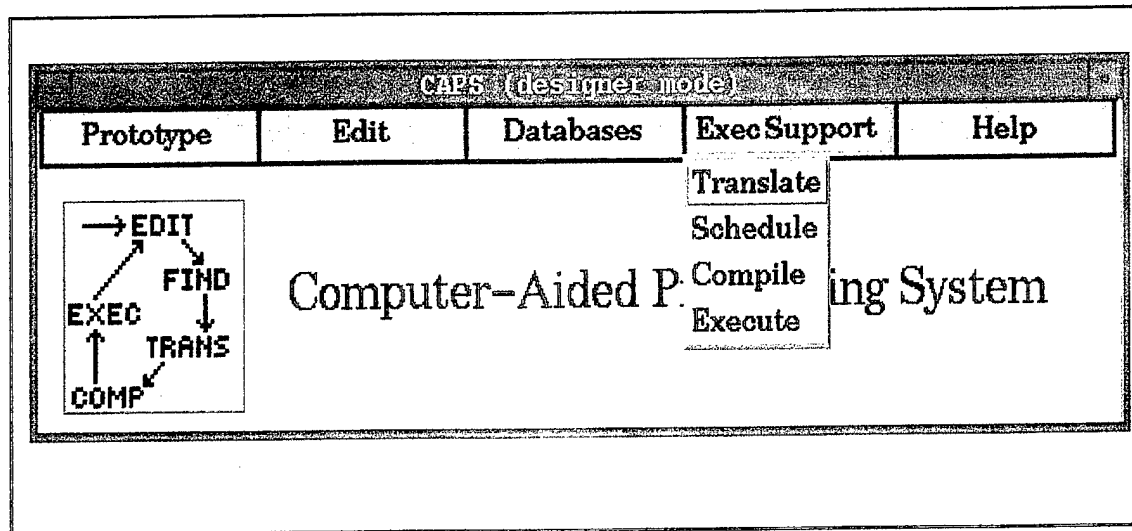


Figure 2.17. Execution Support pull-down menu.

Table 2.6. Execution Support Menu Command Buttons and Functions.

Menu Command Button	Function
<i>Translate</i>	<i>Translates the PSDL language to Ada code.</i>
<i>Schedule</i>	<i>Creates schedules and corresponding Ada code.</i>
<i>Compile</i>	<i>Compiles all the *.a files for the currently open prototype in the designer's private work space.</i>
<i>Execute</i>	<i>Runs the currently open prototype.</i>

## H. USING THE EDITORS

### 1. What Is an Editor?

In general, an editor is a program that enables the user to create and manipulate files. Each development environment provides one or more editors that are useful for program coding, data entry, and documentation [Ref. 14]. Most editors are known as text editors. According to *Random House Personal Computer Dictionary*, *The User's essential accessories Pocket Dictionary*, and the *PC Novice Winter 1996* issue all support this idea.

The tools in the CAPS development environment were developed mainly using five text editors and one graphic editor. The text editors include the PSDL editor, an Ada editor, an interface editor, a requirements editor, and a change request editor. The graphic editor is known as the Graphic Editor.

## 2. PSDL Editor

The PSDL editor allows the designer to create, edit and view the design representations used by CAPS. As stated in the master thesis *Enhancements For The CAPS Prototyping System Description Language Syntax-Directed Editor*, the PSDL editor consist of three parts: the syntax-directed editor (SDE), the Graphic Editor, and the Graph Viewer. The table below gives the function of each part.

**Table 2.7. PSDL Parts and Functions.**

Part	Function
SDE	It has the capability to verify each program line entered to assure that it is syntactically correct, without delaying entry. Reduces the time needed to correct compiling errors; and speeds up the compiling process [Ref. 14].
Graphic Editor	It displays the created or modified data flow design representations; allows the user to enter timing control constraints.
Graph Viewer	It displays the created or modified prototype that was built in the Graphic Editor.

Figure 2.19 displays the PSDL editor. This editor interface includes such user interface elements such as a title bar, three window menu icons, a menu bar labeled with menu names that have pull-down menus associated with them, a work area which is divided into three smaller windows which are labeled as such, two scroll bars located on the left of the first and second windows which are labeled, and two separator lines that divides the larger

window into smaller ones.

**NOTE:** Only different user interface elements stand out in the illustration with text and shadowed boxes.

***a. Title Bar***

The title bar labels the name of the window (`psdl_editor`) followed by a colon to separate the window name from the open prototype file. The “.psdl” is an extension at the end of the open prototype file to help the user identify that the file is PSDL file created using the PSDL language.

***b. Window Menu Icons***

The window menu icons are the standard icons for the Unix environment.

***c. Menu Bar***

This menu bar includes such names as CAPS-Cmds, File, Edit, View, Tools, Options, Structure, Text, and Help. The help in this interface is context sensitive (Not just an informative dialog box). Each of these menu names have pull-down menus associated with them that pulls down on the screen when the menu item is selected.

***d. Divided Windows***

This interface is divided into three smaller windows. Each window displays a shadowed box and a label indicating the window. Window 1 shown in Figure 2.18 displays an action word followed by a filename indicating the file the action was performed on. Window 2 shows the PSDL language source code based on the data that is entered in the Graphic Editor. Notice the all capital letter words, they are known as keywords used in the PSDL grammar. Window 3 list the PDSL component and the menu associated with it.

*e. Scroll Bars*

The scroll bars are known as vertical scroll bars. They are located on the left side of window 1 and window 2. They allow only up and down movement to display data on the screen.

*f. Separator Lines*

This is the horizontal object with a solid filled square box, located near the right side of the window. The square box serves as a handle to adjust the size of a window.

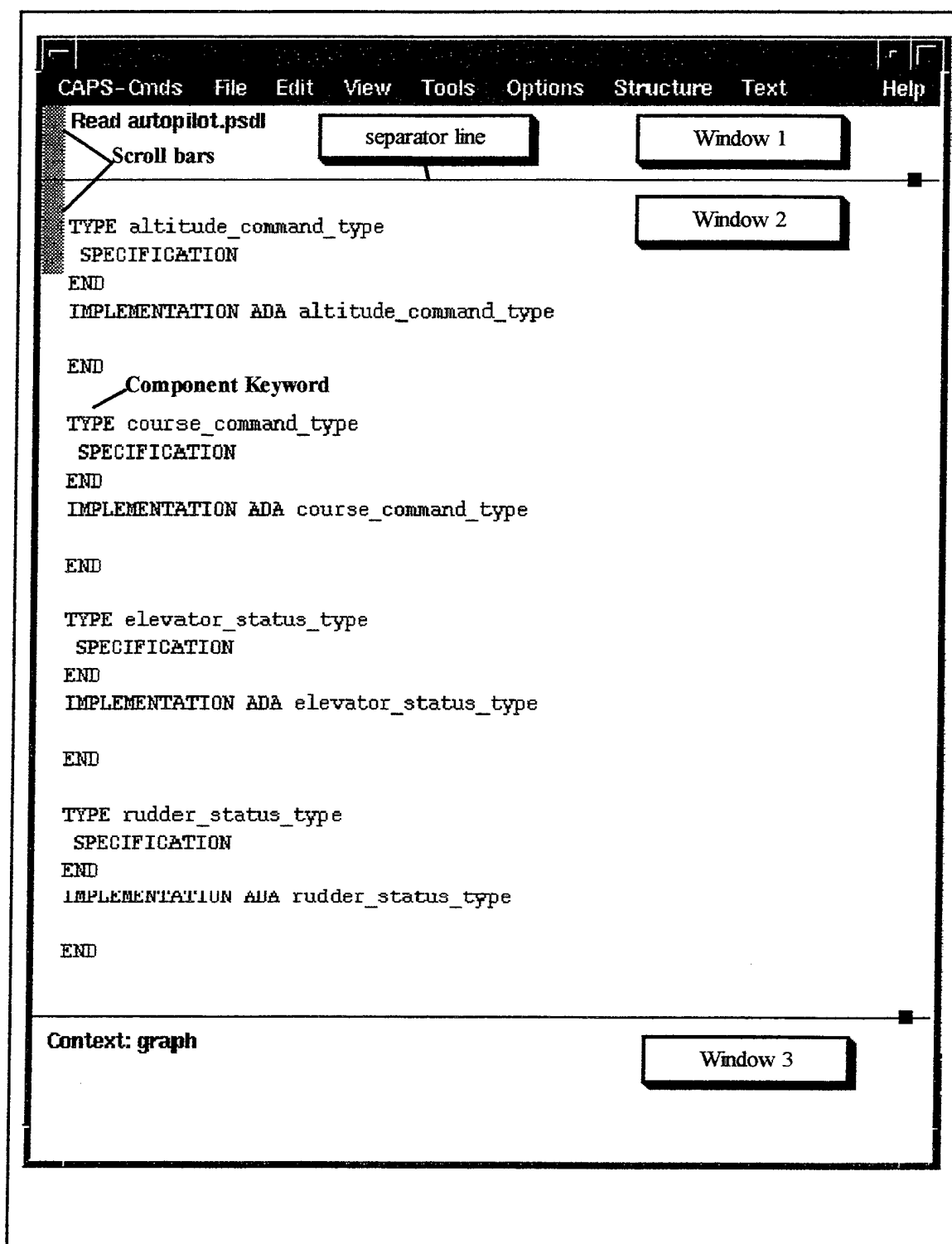


Figure 2.18. PSDL editor interface.

### 3. Graphic Editor

The graphic editor allows the user to draw graphical representations of real-time prototypes and allows modifications to existing data flow diagrams or computational graphs. This is a subsystem of the PSDL editor. The Graphic Editor is further detailed in the Graphic Editor Reference Manual. The Graphic Editor interface is also shown in this manual.

### 4. Ada Editor and Other Text Editors

The Ada editor is one of the text editors which allow designers to view and edit Ada code. It is the CAPS System Defaults option available via selection from the Edit menu that allows the user to select either the Ada SDE or one of the other text editors (vi or emacs). Figure 2.19 shows the CAPS System Defaults dialog box.

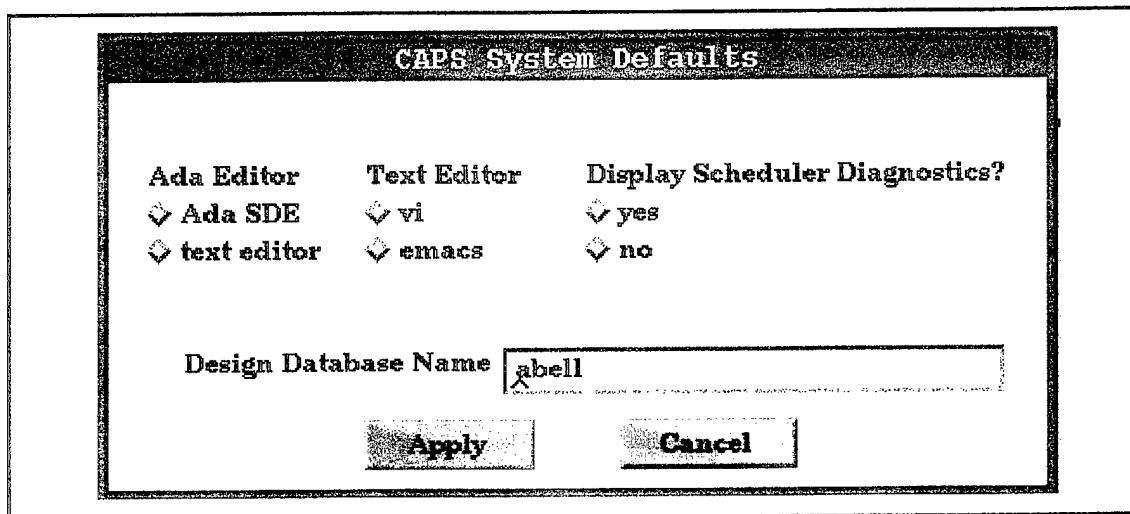


Figure 2.19. CAPS System Defaults dialog box.

Figure 2.20 displays the "CAPS Edit Selection" dialog box which allows the user to edit Ada files. The dialog box appears on the screen as a result of positioning the mouse pointer on the Edit menu name then selecting the Ada menu button. More information about



the Ada option is located in an informative CAPS Help dialog box. To view this information the user must first select the Help button in the “CAPS (designer mode)” window then select the Edit menu name.

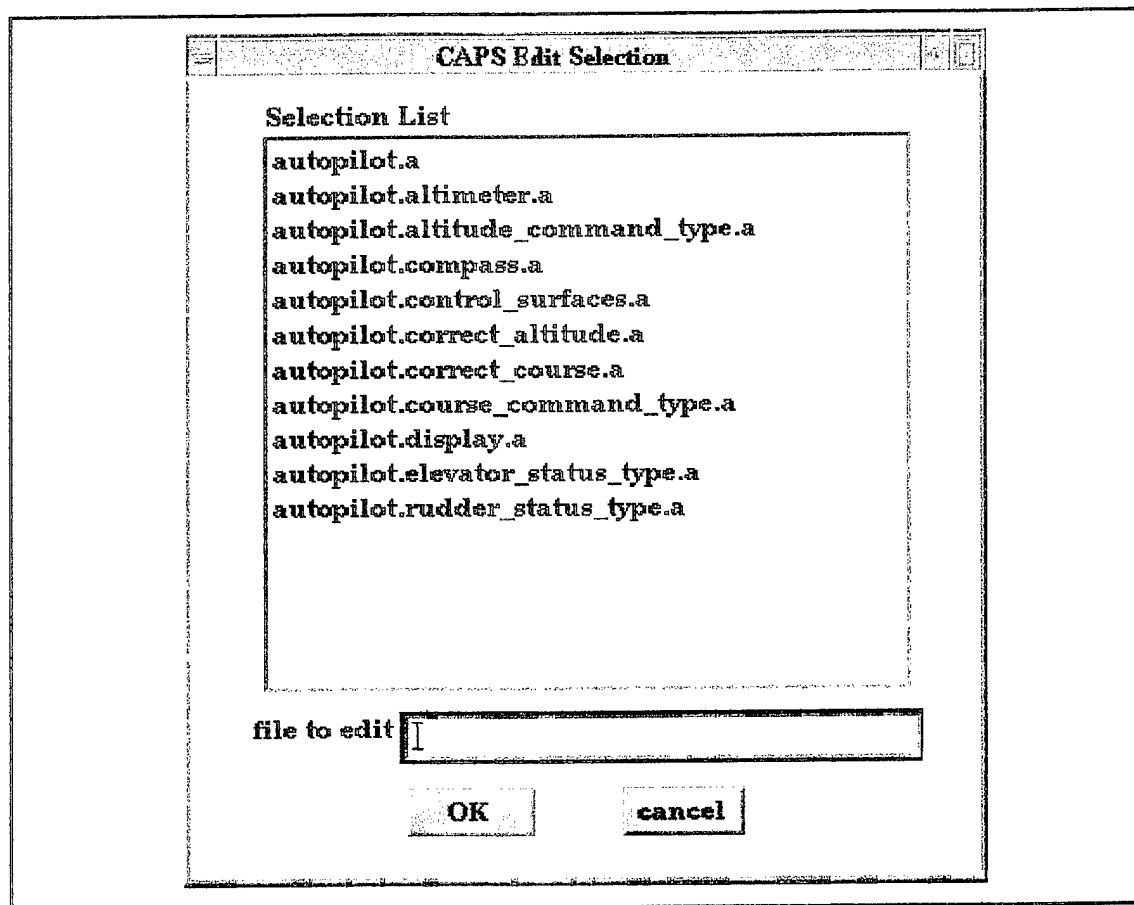


Figure 2.20. CAPS Edit Selection, the result of selecting Ada from the Edit menu.

## 5. Interface Editor

The interface editor allows the user to interactively construct a graphical user interface for the prototype. Often TAE (Transportable Applications Environment) plus is used. According to [LQ92], TAE plus is based on X Windows. When selecting the Interface menu button from the Edit menu, Figure 2.21 is one of the elements displayed on the screen. Figure 2.22 is the other interface that appears on the screen.

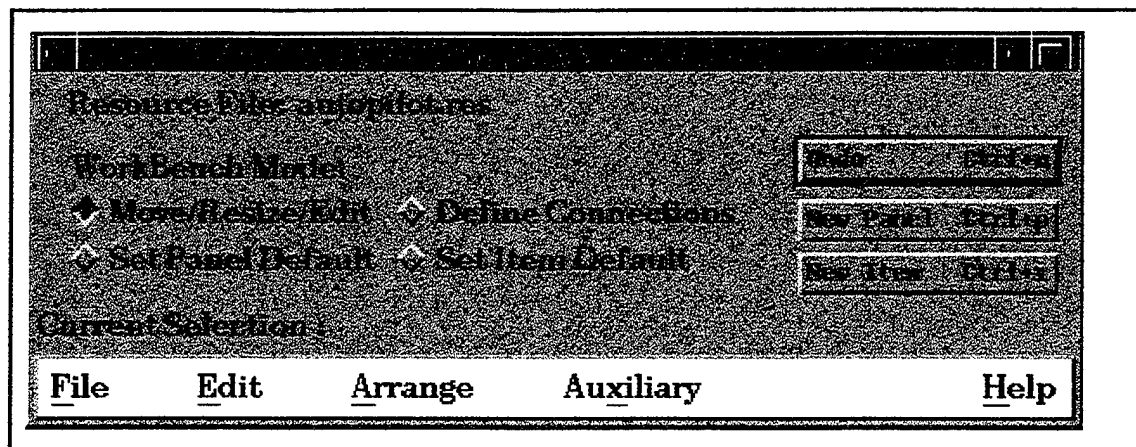


Figure 2.21. TAE Plus Workbench V5.3 interface, the result of selecting the Interface button from the Edit menu. See Figure 2.15.

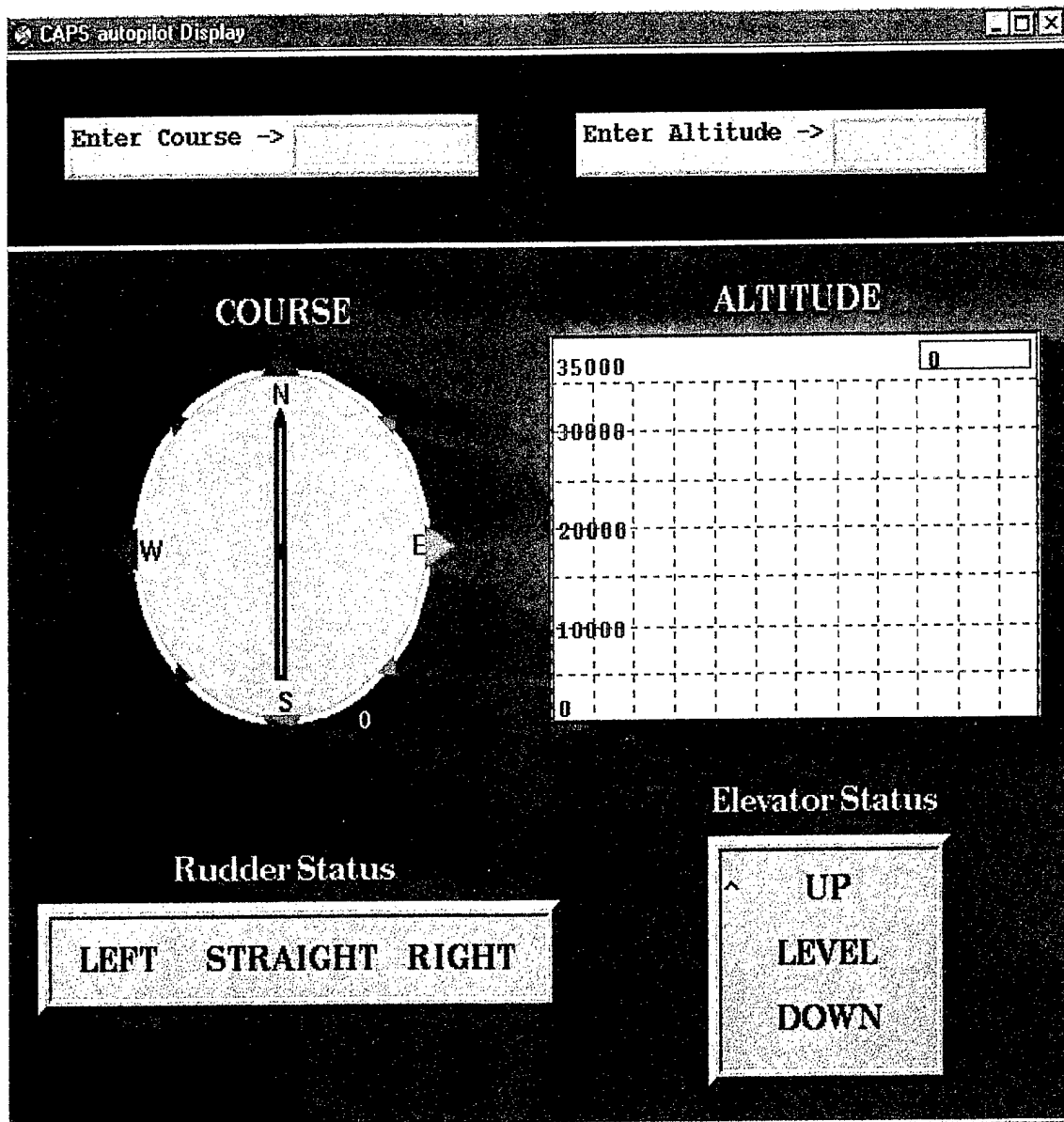


Figure 2.22. CAPS autopilot Display TAE Plus interface, also the result of selecting the Interface menu button from the Edit menu.

## 6. Requirements Editor

The Requirements editor allows the user to view and edit requirements descriptions. Figure 2.23 shows the Requirements interface. This interface appears on the screen by selecting the Requirements menu button from the Edit menu shown in

Figure 2.15. The Requirements interface is simply a dialog box which allows the user to enter data or modify data by using the text box dialog element.

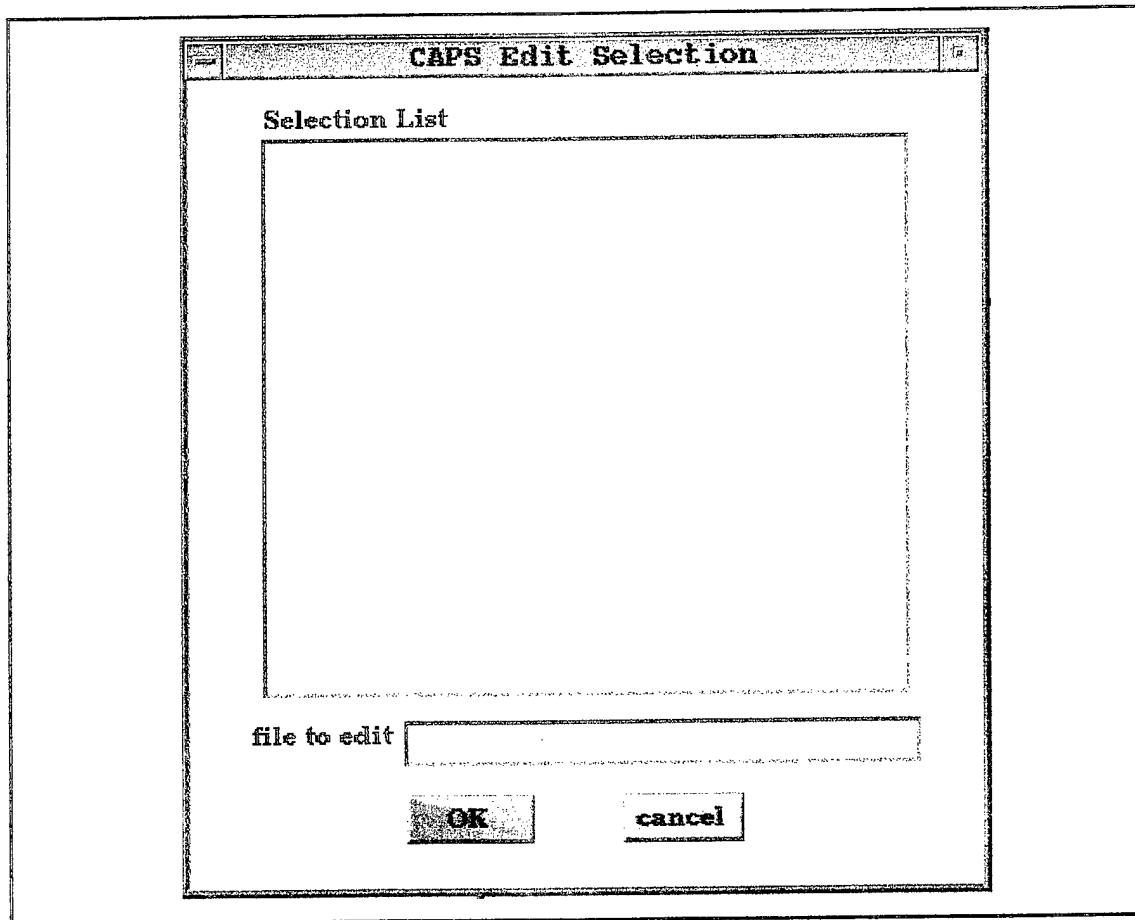


Figure 2.23. Requirements interface.

## 7. Change Request Editor

The Change Request editor also allow users to enter or modify change requests in response to prototype demonstrations. The same illustration shown in Figure 2.23 appears on the screen if the selection list is empty. If any data has been entered into the list then it is displayed in the selection list when it appears on the screen. This interface is accessible by selecting the Change Request menu button from the Edit menu (Figure 2.15).

## **I. USING THE EXECUTION SUPPORT SYSTEM**

The execution support system is one of CAPS subsystems. From Figure 2.17, this subsystem is accessed by positioning the mouse pointer over the Exec Support menu name and then clicking the SELECT mouse button. The Execution Support System has three software tools:

### **1. Translator**

The Translator generates code that binds reusable components extracted from the software base if they are found; or it generates code that binds the components that are custom-built.

### **2. Scheduler**

The Scheduler generates two types of schedules: a high-priority *Static Schedule* and a low priority *Dynamic Schedule*.

- The static schedule allocates time slots for time critical operators (with strict deadlines).
- The dynamic schedule invokes non-constrained operators during execution time slots not previously allocated.

### **3. Compiler**

The Compiler compiles generated Ada programs. CAPS uses the Sun Ada compiler. This manual does not explain the Sun Ada compiler operation.

**NOTE:** Prototypes must be successfully translated and scheduled before compilation occurs. To access any one of the previous mentioned tools, see Figure 2.20. **Select** the appropriate menu button to perform the desired task from the Exec Support menu.

## **J. USING THE PROJECT CONTROL SYSTEM**

As pointed out in [Ref. 11], the project control system can be considered an automated management tool that is most likely to be used by project managers. As the size of a project increases and more people are assigned to the project, it normally means that more time will be spent on communicating to team members and less time spent analyzing, designing, and developing. These large and often complex projects dictates longer time to complete due to personnel turnover. To better manage such problem, all documentation is stored and managed on-line. One system used to accomplish such is the project control system. It consists of two software tools:

### **1. Evolution Control System**

The Evolution Control System (ECS) is designed to give automated help to the task of coordinating concurrent efforts of prototype design team(s) and managing the multiple design versions that are produced.

### **2. The Merger**

The Merger helps to combine the product of two or more independently developed prototype changes, thereby facilitating parallel enhancements and applying common changes to multiple versions.

**NOTE:** The Project Control System is not implemented in CAPS release 1.1.

## **K. USING THE SOFTWARE BASE**

Several sources [Ref. 1, Ref. 3, Ref. 4, and Ref. 5] all describe the software base. The first reference informs the researcher that the software base is part of the software database system. It further states that "the software base provides reusable software components for realizing given PSDL specifications." The second reference illustrates this concept. The third reference gives an example of the software base being used as a repository for reusable software parts with search engine. Finally, the fourth reference gives the clearer, concise, and simple definition; the software base is a CAPS tool that keeps track of the PSDL description and Ada implementations for all reusable software components in CAPS.

**NOTE:** The Software Base is not currently implemented in the CAPS Release 1.1 version.

### **III. CAPS GRAPHIC EDITOR REFERENCE MANUAL**

In this chapter, the thesis provides the outline format for creating the CAPS Graphic Editor Reference Manual. The format is similar to that of the CAPS User's Interface Reference Manual. Most of the same resources were researched and reviewed and found to be applicable to use in designing, developing, and creating this manual.

The significant CAPS tool discussed in this manual is the Graphic Editor. The accurate application and usage of the user interface elements in this tool is essential to prototyping validation process. This manual too included in its illustrations of the interfaces that CAPS users encounter and must interact and manipulate to execute the CAPS tools used in the Graphic Editor to create or modify an existing prototype design.

Finally, the last element in this manual is the organization of the sections which includes the visual, graphical, and textual contents of the CAPS Graphic Editor Reference Manual. This reference manual is organized into eight sections.

#### **A. GRAPHIC EDITOR PREFACE**

##### **1. About This Manual**

This manual contains useful information in creating and editing prototype designs using the Graph Editor and its associated tools.

##### **2. What This Reference Manual Does**

This manual describes the Graphic Editor. It also, gives instructions on how to invoke the Graphic Editor and how to use its user interface elements. Finally, it states and illustrates the user interface elements and their usage within the Computer-Aided Prototyping System



(CAPS) software development environment.

### **3. Audience**

To effectively use the Graphic Editor tool, the designers and CAPS users should know how to perform basic user interface operations such as:

- Choosing items from a menu bar
- Choosing items from a tool sidebar
- Closing windows
- Deleting objects from a graph
- Displaying pull-down menus
- Entering text in a text box
- Invoking CAPS tools such as the Graphic Editor
- Selecting exclusive choice items in a dialog box
- Selecting items in a scroll list
- Selecting non-exclusive choice items in a dialog box
- Using the scroll bars to view data
- Using the different drawing tools to create and modify prototype designs

If you are not familiar with these operations, refer to a *Microsoft Windows User's Guide* or *Motif User's Manual* for more details and how to use buttons, menus, scrolling list and selection list. Also , see the *CAPS Tutorial* for information on using the Graphic Editor.

### **4. How This Manual Is Organized**

This manual organizes its sections in the following manner:

- **SECTION A-GRAPHIC EDITOR PREFACE**-Provides an introduction to the Graphic Editor Reference Manual.
- **SECTION B-GETTING STARTED**-Includes instructions on how to invoke the Graphic Editor, how to execute CAPS, how to invoke CAPS with a new prototype and how to invoke CAPS with an existing prototype.
- **SECTION C-GRAPHIC EDITOR**-Describes the Graph Editor window, gives its usage within CAPS, illustrates its user interface elements and provides instructions on how to create, edit, save, and print a graph. Instructions on how to close the Graph Editor is also included.
- **SECTION D-GRAPH EDITOR INTERFACE**-Labels, describes, and gives instructions on how to display the graph editor window, and illustrates its user interface elements.
- **SECTION E-USING THE APPLICATION ICON OR WINDOW MENU BUTTON**-Points out the similarity and difference between the application icon and the window menu button.
- **SECTION F-THE GRAPHIC EDITOR DRAWING SPACE**-Displays, labels, and describes the drawing space. Also shows its user interface elements and gives instructions on their usage within the CAPS environment.
- **SECTION G-THE MENU BAR AND PULL-DOWN MENUS**-Shows the names of the pull-down menus used within the Graph Editor interface.
- **SECTION H-THE TOOL SIDEBAR**-Describes the tool sidebar, explains its drawing tools, and illustrates their usage within the CAPS environment.

## **B. GETTING STARTED**

### **1. Invoking the Graphic Editor**

Before creating and editing any prototype design, **ensure** that you can access the CAPS software development environment. See the CAPS User's Manual if assistance is required. First, execute CAPS then, invoke the Graphic Editor.

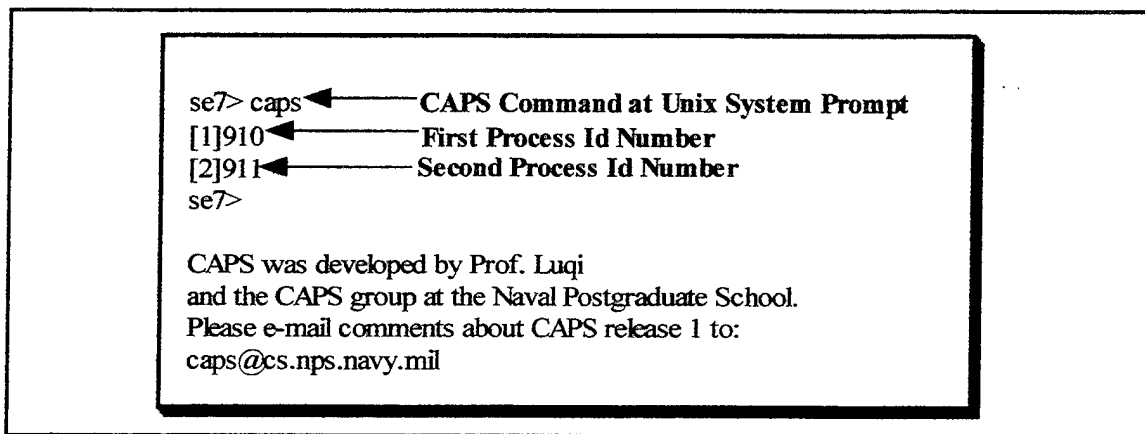
## 2. Executing CAPS

To execute the CAPS program follow the steps listed below.

**Step 1:** Type *caps* at the system prompt.

**Step 2:** Press the Enter or Return key on the keyboard.

**Step 3:** Notice that two process id numbers followed by a Unix system prompt followed by some CAPS information is displayed on the screen. The contents of an “xterm” execution window after CAPS has been started illustrated in the shadowed box shown in Figure 3.1.



```
se7> caps  
[1]910  
[2]911  
se7>
```

Annotations:

- se7> caps ← CAPS Command at Unix System Prompt
- [1]910 ← First Process Id Number
- [2]911 ← Second Process Id Number

CAPS was developed by Prof. Luqi  
and the CAPS group at the Naval Postgraduate School.  
Please e-mail comments about CAPS release 1 to:  
caps@cs.nps.navy.mil

**Figure 3.1.** An example of information displayed in an “xterm” execution window after CAPS has been started.

There are two ways to invoke the Graphic Editor. It can be invoked either by a new prototype or by an existing prototype.

**Step 4:** Decide whether the Graphic Editor will be invoked by a new or an existing prototype.

**Step 5:** Wait for the “CAPS (designer mode)” User Interface window to appear on the screen as shown in Figure 3.2. The user interface elements shown will be discussed in later chapters.

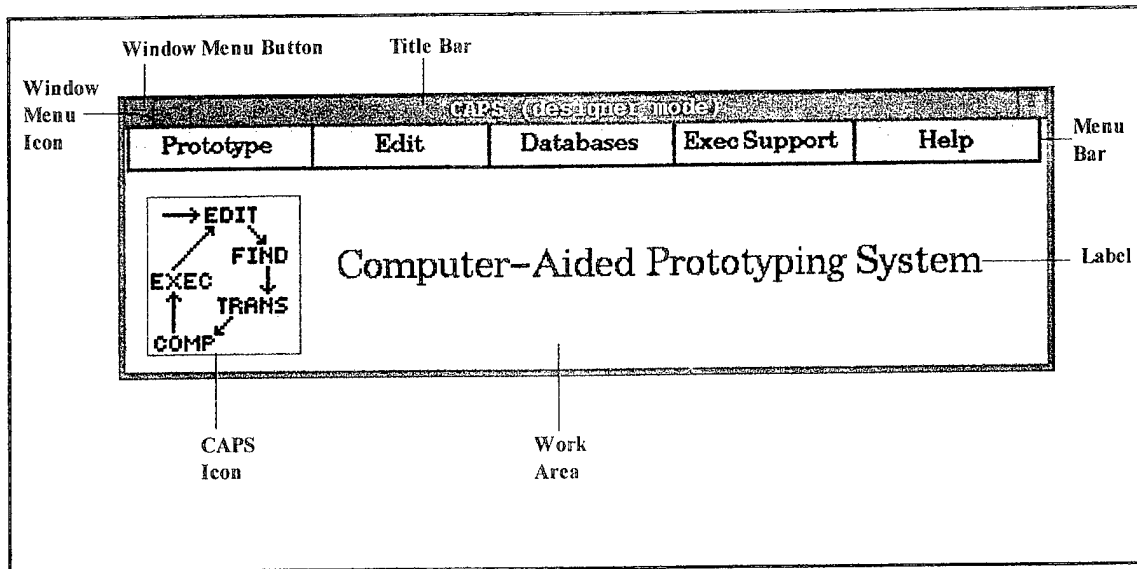


Figure 3.2. CAPS (designer mode) interface, the link to executing CAPS tools.

**Step 6:** Select the Prototype menu item from the menu bar by **positioning** your mouse pointer on/over the Prototype label.

**Step 7:** Click the SELECT mouse button. The Prototype pull-down menu appears on the screen just below the Prototype label as shown in Figure 3.3. Its default choice highlighted with a hollow rectangular outline.

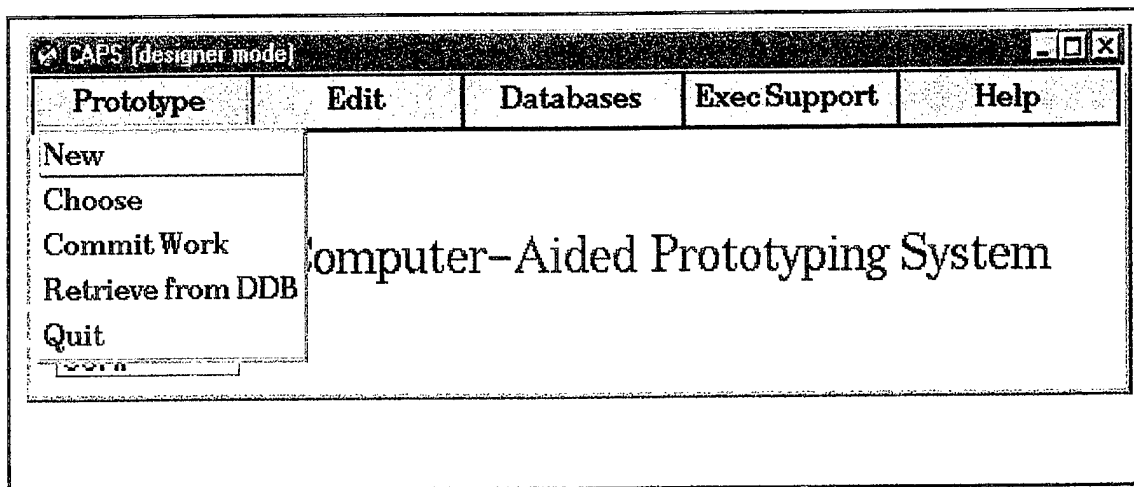


Figure 3.3. Prototype pull-down menu, after executing CAPS.

**Step 8:** Follow the steps in either Invoking the Graphic Editor with a New Prototype or Invoking the Graphic Editor with an Existing Prototype section to invoke the Graphic Editor with the type of design you decided.

### 3. Invoking the Graphic Editor with a New Prototype

This section provides the steps to invoke the Graphic Editor with a new prototype.

**Step 1:** From the Prototype menu, which is shown in Figure 3.2, either **press** the Enter or Return key on the keyboard to select the default choice or **position** the mouse pointer on/over the menu item with the New label.

**Step 2:** Click the SELECT mouse button to select the New option button as your selected choice. A “new prototype” dialog box appears on the screen as shown in Figure 3.4.

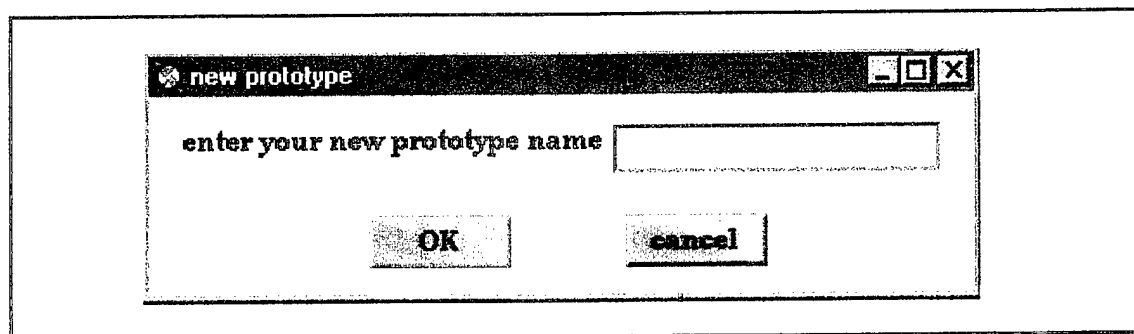


Figure 3.4. New Prototype dialog box.

**Step 3:** Click inside the “enter your new prototype name” text box. A flashing vertical insertion point (I-bar) cursor appears in the text box to indicate the point at which the user can start entering text.

**NOTE:** When positioned over a text box, the mouse pointer takes the shape of an I-beam. To edit existing information in the box, click the I-beam at the location where you want to make an insertion or a change.

**Step 4:** Enter the name you wish to label as the new prototype. Either **position** the mouse pointer over the green OK button and click the SELECT mouse button or **press** the Enter or Return key on the keyboard for the process to accept the data that has been entered.

Two windows pop up on the screen. One is the syntax-directed editor (SDE) window. The SDE is known as the “PSDL editor” because of its use of the PSDL language. The other window is the Graph Viewer. It is used only for viewing. It cannot be modified.

In the previous chapter Figure 2.18 shows the SDE window which is labeled as the “psdl editor” window. Also, **notice** that the name of the open prototype appears in the title bar following the colon and the title of the window.

Figure 3.5 that is shown below is the Graph Viewer Window and its user interface elements. This window is used to view updates made to a graph. It is not modifiable.

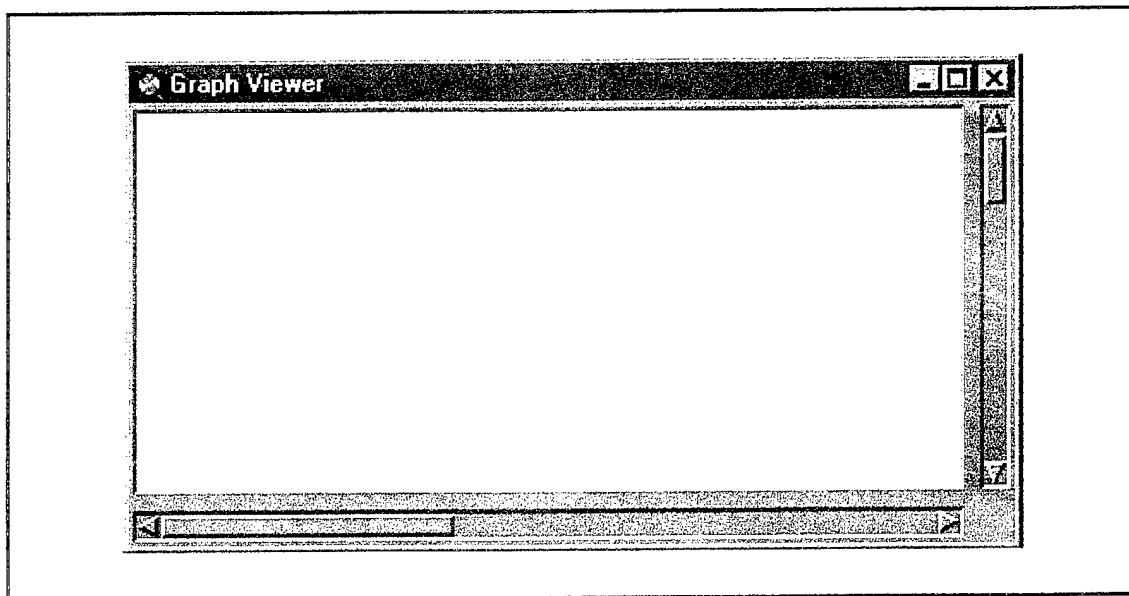


Figure 3.5. Graph Viewer window.

**Step 5:** In the “psdl\_editor” window, position the mouse pointer over the CAPS-Cmds label in the menu bar. This menu item is highlighted.

**Step 6:** Click the SELECT mouse button to pull down the CAPS-Cmds pull-down menu.

**Step 7:** Notice that only an outline of the pull-down menu flashes on the screen and then disappears off the screen.

**Step 8:** With the mouse pointer still positioned over the CAPS-Cmds label in the menu bar **hold down** the SELECT mouse button for the menu to stay on the screen.

Figure 3.6 shows the “psdl\_editor” with the CAPS-Cmds pull-down menu appearing just beneath the CAPS-Cmds label.

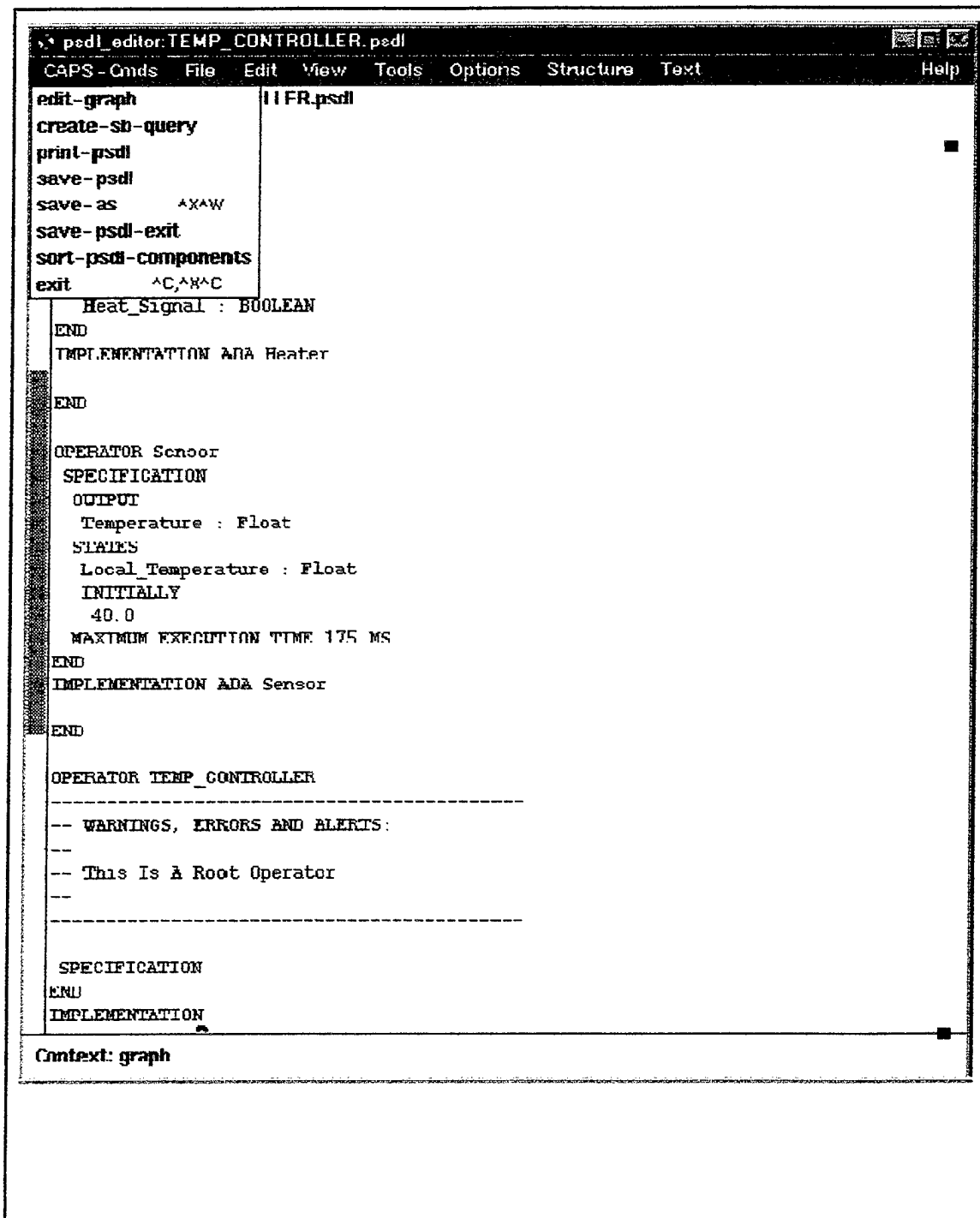


Figure 3.6. CAPS Commands pull-down menu.



**Step 9:** From the CAPS-Cmds pull-down menu, **position** the mouse pointer on/over the edit graph label. The menu item appears highlighted by reverse video.

Reverse video means that background and foreground switch appearance. For example, if the background is dark and the foreground is light then reverse video is now the background appears light and the foreground appears dark.

**Step 10:** Click the SELECT mouse button, the Graph Editor window appears on the screen. The Graphic Editor has been invoked and now is displayed on the screen.

#### **4. Invoking the Graphic Editor with an Existing Prototype**

This section provides the steps required to invoke the Graphic Editor with an existing prototype.

**Step 1:** Access the Prototype pull-down menu from the menu bar as shown in Figure 3.6 by positioning the mouse pointer on/over the Prototype label.

**Step 2:** Click the SELECT mouse button. The Prototype pull-down menu appears on the screen just below the Prototype label.

**Step 3:** **Position** the mouse pointer on/over the Choose button to highlight it. The hollow rectangular outline indicates that the button has been highlighted and is your desired selection.

Figure 3.7 shows the Prototype pull-down menu with the Choose option button highlighted with a hollow rectangular outline.

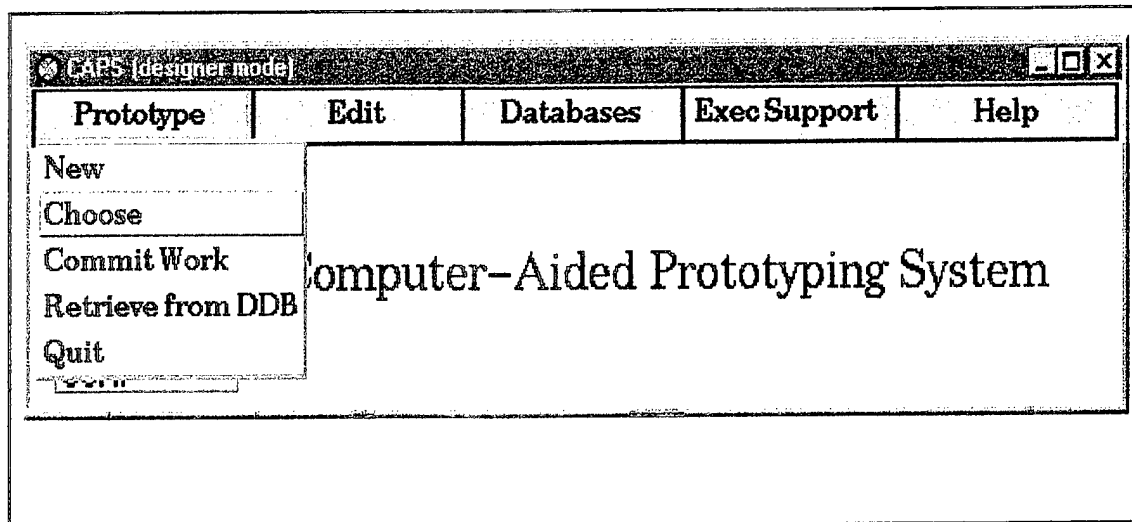


Figure 3.7. Prototype pull-down menu.

Step 4: After selecting the Choose button, an “open prototype” dialog box with a Select Prototype selection list appears on the screen which is shown in Figure 3.8.

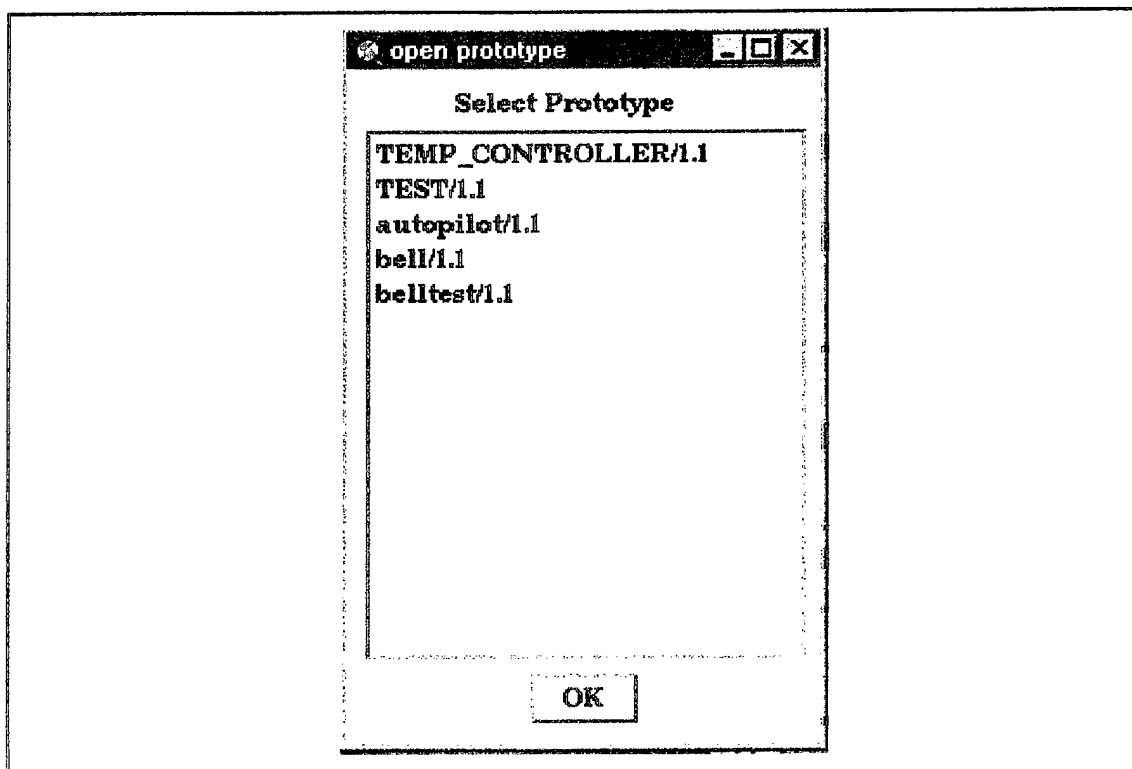


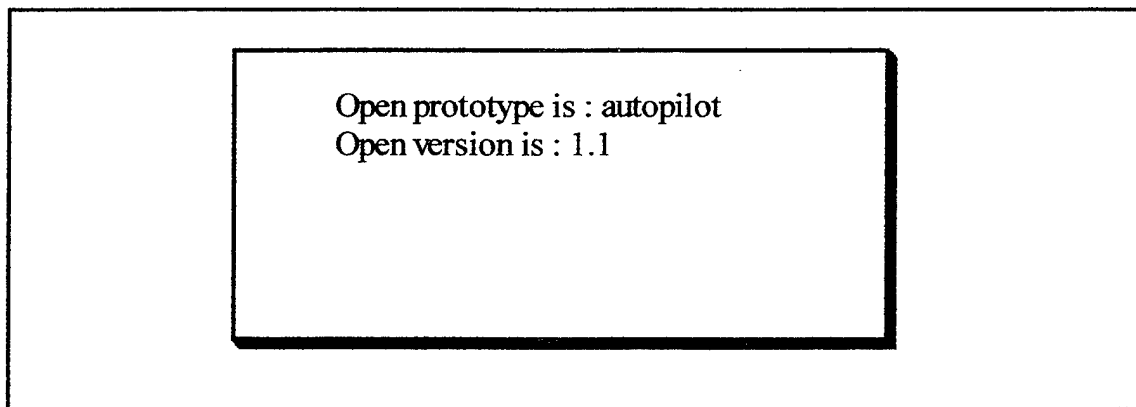
Figure 3.8. Open Prototype dialog box.

**Step 5: Position** the mouse pointer over the name label of the prototype you wish to select. A solid rectangle box is displayed to highlight your choice.

**Step 6: Position** the mouse pointer over the OK button located at the bottom of the dialog box for the system to accept your choice.

**Step 7: Notice** the “xterm” execution window. It displays a message giving the current state of CAPS.

Figure 3.9 displays the message that is displayed in the “xterm” execution window.



**Figure 3.9. An xterm window recording after opening a prototype.**

**Step 8: Position** the mouse pointer over the Edit menu item in the “CAPS (designer mode)” window. The result of this action is shown in Figure 3.10.

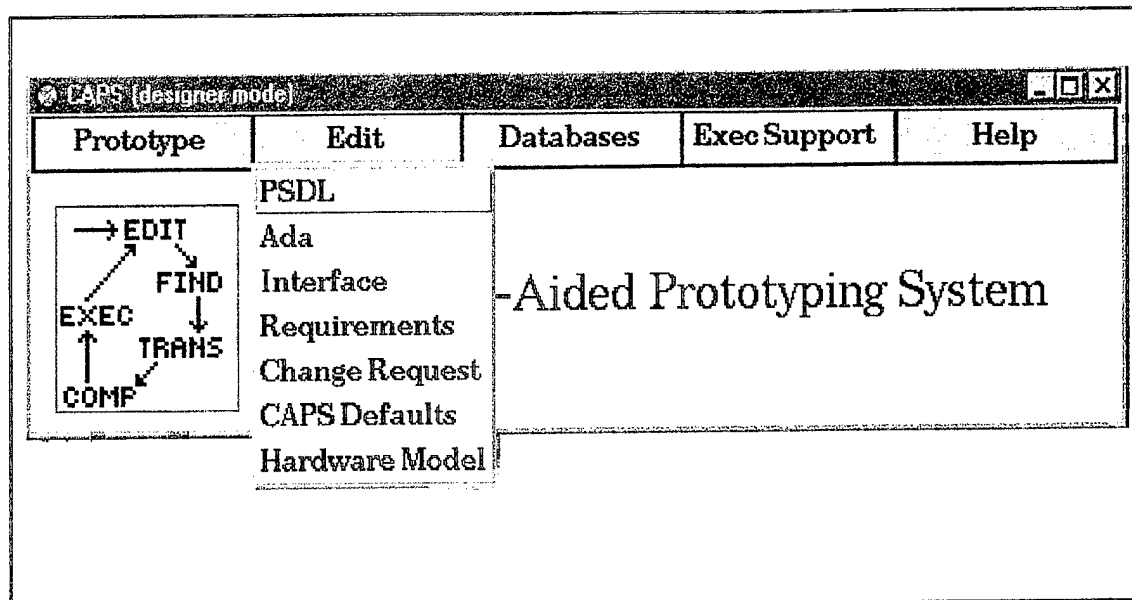


Figure 3.10. Edit Pull-Down Menu.

Step 9: From the Edit menu, position the mouse pointer over the highlighted choice or press the Enter or Return key on the keyboard.

Step 10: Click the SELECT mouse button to select the highlighted choice (PSDL). The "PSDL editor" and the Graph Viewer windows appear on the screen.

Step 11: See the previous section and follow steps 5 - 10. Since at this point, the steps are the same for invoking the Graphic Editor.

### C. GRAPHIC EDITOR (GE)

This chapter describes the Graphic Editor, states its usage within CAPS, and lists the user interface elements in its main window, the Graph Editor.

#### 1. Description

The Graphic Editor is one of CAPS interface tools. It has a set of interactive, graphical drawing tools, that are used to create, or modify PSDL graphs. The PSDL graphs are annotated data flow diagrams (graphical representations of software designs) showing flow of data.

## **2. Usage within CAPS**

The primary function of this tool is to draw, display, and edit PSDL graphs in the drawing space. The drawn software designs models the graphical representation of a software system as a network of operators communicating via data streams. The Graphic Editor also, allows the user to interact with other CAPS processes, interface with other software tools, and manipulate drawn PSDL graphs.

## **3. User Interface Elements**

This section lists the Graphic Editor's user interface elements. The Graph Editor window is its primary interface element. The editor displays the following listing:

- an application icon in the upper left corner of the window, if CAPS was executed from a PC,
- a window menu button in the upper left corner of the window, if CAPS was executed from a Unix workstation,
- a tool sidebar which contains five buttons,
- a menu bar with three menu items each which has a pull-down menu,
- a drawing space,
- two scroll bars (horizontal and vertical) with direction arrows and a scroll box,
- a status bar which has two text boxes,
- and a dialog box is popped up on the screen when the Properties or Stream button is clicked.

#### 4. Creating a Graph Using the Graphic Editor

To create a graph within the Graphic Editor of CAPS the first step is to run CAPS. After starting CAPS, the Graphic editor must be invoked. See the Getting started and Invoking the Graphic Editor sections for more details and assistance.

#### 5. Steps to Create a PSDL Graph Using the Graphic Editor

Figure 3.11 shows the drawing tools and icons.

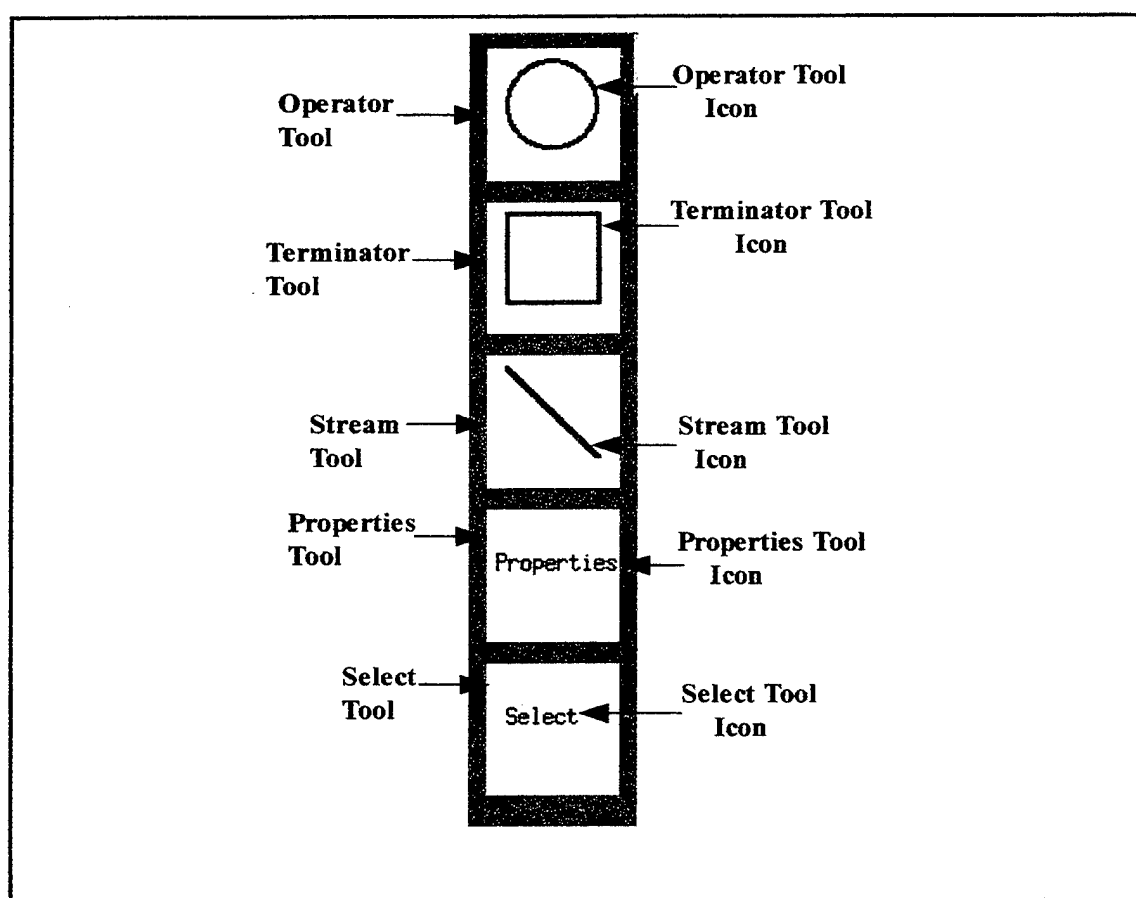


Figure 3.11. Graphic Editor drawing tools and icons.

**Step 1:** Select the Operator Tool to draw an operator or select the Terminator Tool to draw a terminator or select the Stream Tool to draw a data stream.

**Step 2: Position** the mouse pointer in the drawing space where you want to place the symbol.

**Step 3: Click** the SELECT mouse button, the symbol appears on the screen with the default properties set and is positioned at the position of the mouse pointer.

**Step 4: Repeat** steps 1- 3, until design contains the symbols you want it to have and they are annotated and positioned where you want them.

## **6. Editing a PSDL Graph Using the Graphic Editor**

Again, CAPS must be started and the Graphic Editor must be invoked before editing can take place. See the Getting Started and Invoking the Graphic Editor with an Existing Prototype sections for more details and assistance.

Follow the steps listed below to edit a PSDL graph using the Graphic Editor.

**Step 1: Click** the Select button if the status bar does not display Select Tool. You can identify the current mode by viewing the left text box of the status bar. This text box indicates the active tool.

**Step 2: Use** the scroll bars to view the part of the Graph that is not displayed on the screen.

Other options include can use the Maximize menu button from the application icon pull-down menu or the Maximize menu button from the Window Ops menu to make the screen larger. See the CAPS User's Interface Reference Manual for examples.

**Step 3: Decide** what needs to be changed.

**Step 4: Based** on your decision, to edit the font, color or undelete an object, **click** the Draw Options menu item. A pull-down menu appears on the screen just below the menu item label.

**Step 5: Highlight** your desired choice and then **click** the SELECT mouse button for the system to accept your selection.

**Step 6:** Repeat step 5 for the font and color submenus that appear on the screen.

**Step 7:** When changes are complete, **position** the mouse pointer over the Graph menu item label and click the SELECT mouse button to display its pull-down menu.

**Step 8:** Repeat step 5.

## **7. Saving a Graph within the Graphic Editor**

This section provides the steps to perform to save a graph within the Graphic Editor.

**NOTE:** When the Save menu option is selected from the Graph pull-down menu, the prototype is saved only in temporary memory and is not written to a file.

**Step 1:** **Position** the mouse pointer on/over the Graph menu item in the menu bar.

**Step 2:** **Click** the SELECT mouse button, the Graph pull-down menu appears on the screen just beneath its label.

**Step 3:** **Position** the mouse pointer on/over the Save option; then, click the SELECT mouse button for the system to accept your choice. The graph is saved only in the temporary memory.

## **8. Printing a Graph within the Graphic Editor**

**NOTE:** When the Print menu option is selected from the Graph pull-down menu, the prototype prints a screen dump of what is in view on the screen.

**Step 1:** **Position** the mouse pointer on/over the Graph menu label in the menu bar.

**Step 2:** **Click** the SELECT mouse button, the Graph pull-down menu appears on the screen just beneath its label.

**Step 3:** **Position** the mouse pointer on/over the Print menu option; then, **click** the SELECT mouse button for the system to accept your choice. Only the portion of the graph that is in view is on the screen is printed to the printer.



## **9. Closing the Graphic Editor**

This section provides the steps for closing the Graphic Editor Window.

**Step 1: Position** the mouse pointer on/over the Graph menu label in the menu bar.

**Step 2: Click** the SELECT mouse button, the Graph pull-down menu appears on the screen just beneath its label.

**Step 3: Position** the mouse pointer on/over the Exit menu option; then, click the SELECT mouse button for the system to accept your choice. Control is then returned to the syntax-directed editor (SDE). The "PSDL editor" window is in view on the screen.

**Step 4: Position** the mouse pointer on/over the CAPS-Cmds label. This menu item is highlighted.

**Step 5: Hold down** the SELECT mouse button, the CAPS-Cmds pull-down menu appears on the screen.

**Step 6: Highlight** the Exit menu option, a dialog box appears on the screen.

**Step 7: Position** the mouse pointer over the Exit label, a rectangular box outlines the label to highlight it.

**Step 8: Click** the SELECT mouse button, control is returned to the "CAPS (designer mode)" window.

**Step 9: Position** the mouse pointer over the Prototype label.

**Step 10: Click** the SELECT mouse button, the Prototype pull-down menu appears on the screen just below the Prototype label. A hollow rectangular box outlines its default selection to highlight it.

**Step 11: Position** the mouse pointer on/over the Quit option. A hollow rectangular box outlines it to highlight it.

**Step 12: Click** the SELECT mouse button, the window disappears and the "xterm" execution window displays the message "Quitting CAPS."

## **D. GRAPH EDITOR INTERFACE**

### **1. Description**

The Graph Editor interface is the main window in the Graphic Editor. It contains all the controls (shown as buttons in Figure 3.12) needed to create a new design or edit an existing design.

### **2. Displaying the Graph Editor Interface**

To display the Graph Editor interface, simply **type** "caps" at the host Unix system prompt, then follow the steps in the Getting Started section to invoke the Graphic Editor.

Once the Graphic Editor is invoked, the Graph Editor (Figure 3.12) will be displayed on the screen.

#### ***a. Application Icon or Window Menu Button***

The application icon is a small picture that represents the program that is executed. This picture functions as a button and with a click of the SELECT mouse button, a pull-down menu is displayed on the screen just beneath the icon. The application icon is located in the upper left corner of the window if the executed program is ran on a PC. See Figure 3.10 illustration.

The window menu button is a small square box located in the upper left corner of the window, that indicates CAPS was executed on a Unix workstation. Its behavior is the same as the application icon when the SELECT mouse button is pressed.

Figure 3.13 displays the application icon demonstrating that CAPS was executed on a PC; however, Figure 3.14 shows the window menu button demonstrating that

CAPS was executed on a Unix workstation.

Accessing the application icon and window menu button pull-down menus are discussed in Section E.

***b. Drawing Space***

The drawing space is the area for drawing, editing, and displaying PSDL graphs. It is located to the right of the tool sidebar and just beneath the menu bar. More details is included in Section E.

***c. Menu Bar***

The menu bar is the horizontal section located near the top of the window and to the right of the tool sidebar and just beneath the title bar. It contains three labels known as menu items. See Figure 3.12 illustration. Each menu item has a pull-down menu associated with them. The pull-down menus are discussed in Section G.

***d. Scroll Bars***

There are two scroll bars attached to the Graph Editor window. One bar is located on the right side of the drawing space. The other scroll bar is located on the bottom of the drawing space. These user interface elements has other elements attached to them which speeds up the process of viewing data that is not displayed within the active window in view.

***e. Status Bar***

The status bar is the horizontal section near the bottom of the window. It is composed of two text boxes (left text box and right text box). The left text box is used to

indicate which tool is active. The right text box is used to display the name of the open prototype design. See Figure 3.12 illustration.

*f.      Tool Sidebar*

The tool sidebar is located to the left of the drawing space and contains five buttons (three drawing tools, a Properties button, and a Select button). Each of the drawing tools has an icon which represents a standard symbol that is used to create a PSDL computational graph. The tool sidebar functionality is explained in Section H.

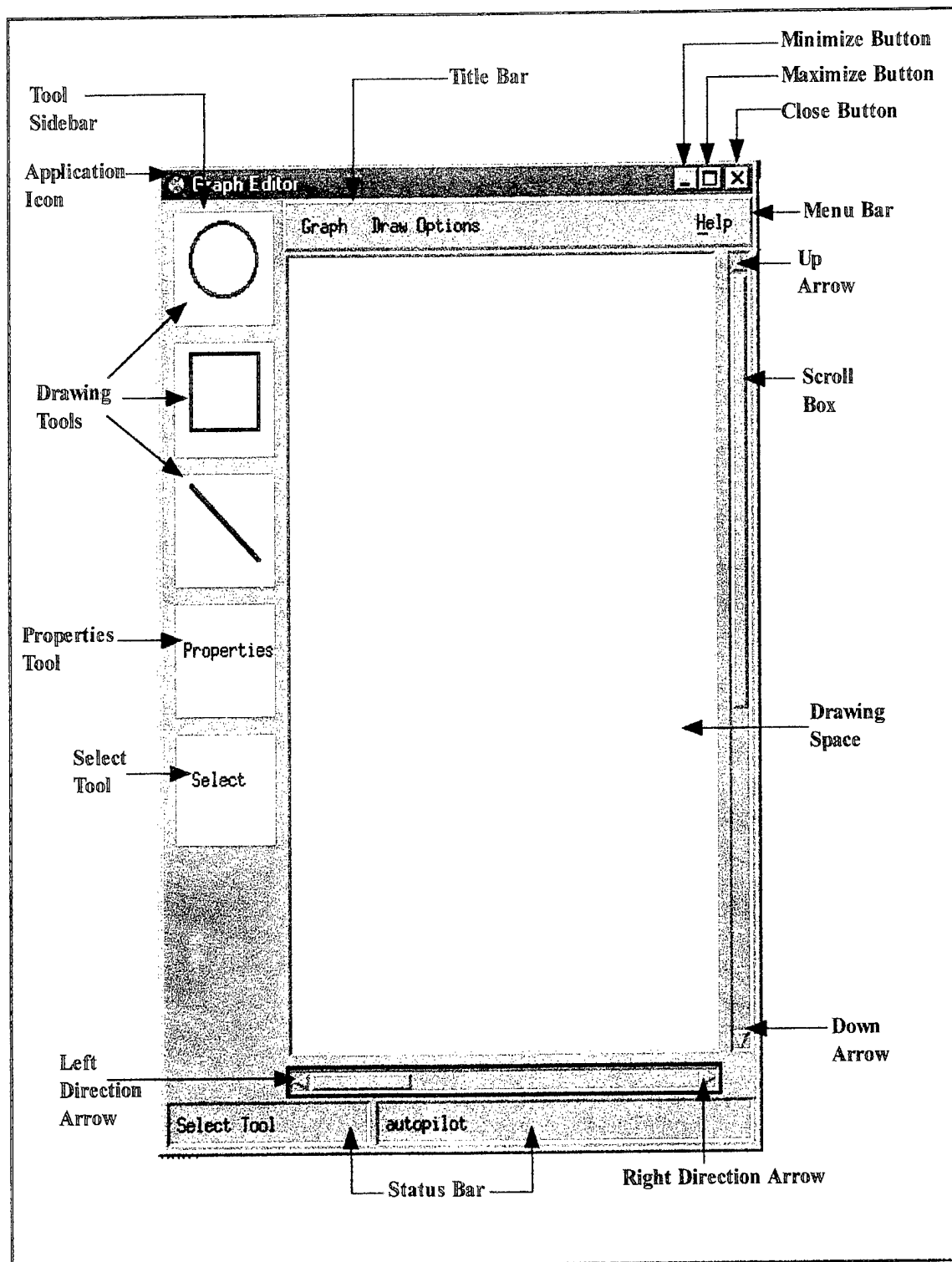


Figure 3.12. Graphic editor main window.

## **E. THE APPLICATION ICON AND WINDOW MENU BUTTON**

### **1. Description**

This Chapter illustrates the application icon which is executed from a PC and a window menu button which is displayed when a Unix work station is used to execute CAPS.

The application icon is displayed at the top of an interface window as a push button with an interactive task linked to it.

### **2. Accessing the Application Icon Menu or Window Menu**

To access the application icon menu or window menu the user would simply position the mouse pointer on/over the object, then you click the SELECT mouse button. A pull-down menu appears on the screen. **Select** the menu button associated with the action to be performed from the pull-down menu.

### **3. User Interface Elements**

The application icon and the window menu button both have pull-down menus which have some of the same functionality of resizing windows. One obvious difference between a PC application icon and a Unix workstation window menu button is the Unix workstation control menu has more options to choose from. See Figure 3.13 and 3.14. Figure 3.14. shows an application icon with its associated pull-down menu indicating that CAP was executed from a PC.

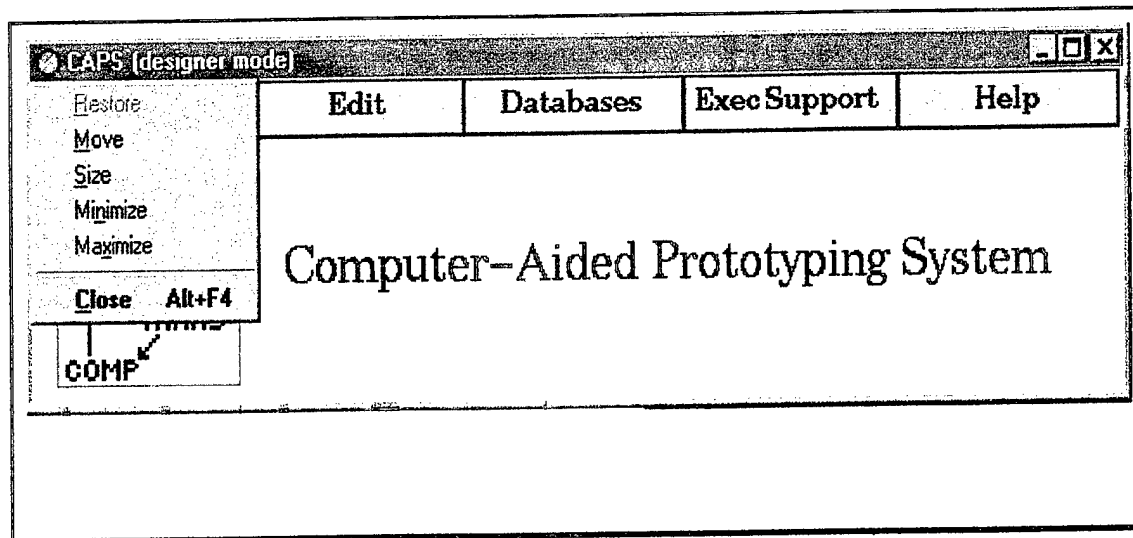


Figure 3.13. PC application icon pull-down menu.

Table 3.1. Application Icon Menu Option Buttons and Functions

<i>Option</i>	<i>Function</i>
<i>Restore</i>	<i>Dimmed when not available. Restores window to its former size after you have enlarged it (by using the Maximize command) or reduced it to an icon (by using the minimize command).</i>
<i>Move</i>	<i>Use the keyboard to move the window to another position.</i>
<i>Size</i>	<i>Use the keyboard to change the size of the window.</i>
<i>Minimize</i>	<i>Reduce the window to an icon</i>
<i>Maximize</i>	<i>Enlarge the window to its maximum size.</i>
<i>Close</i>	<i>Close the current active window.</i>

Figure 3.14 shows a Unix workstation executed window menu.

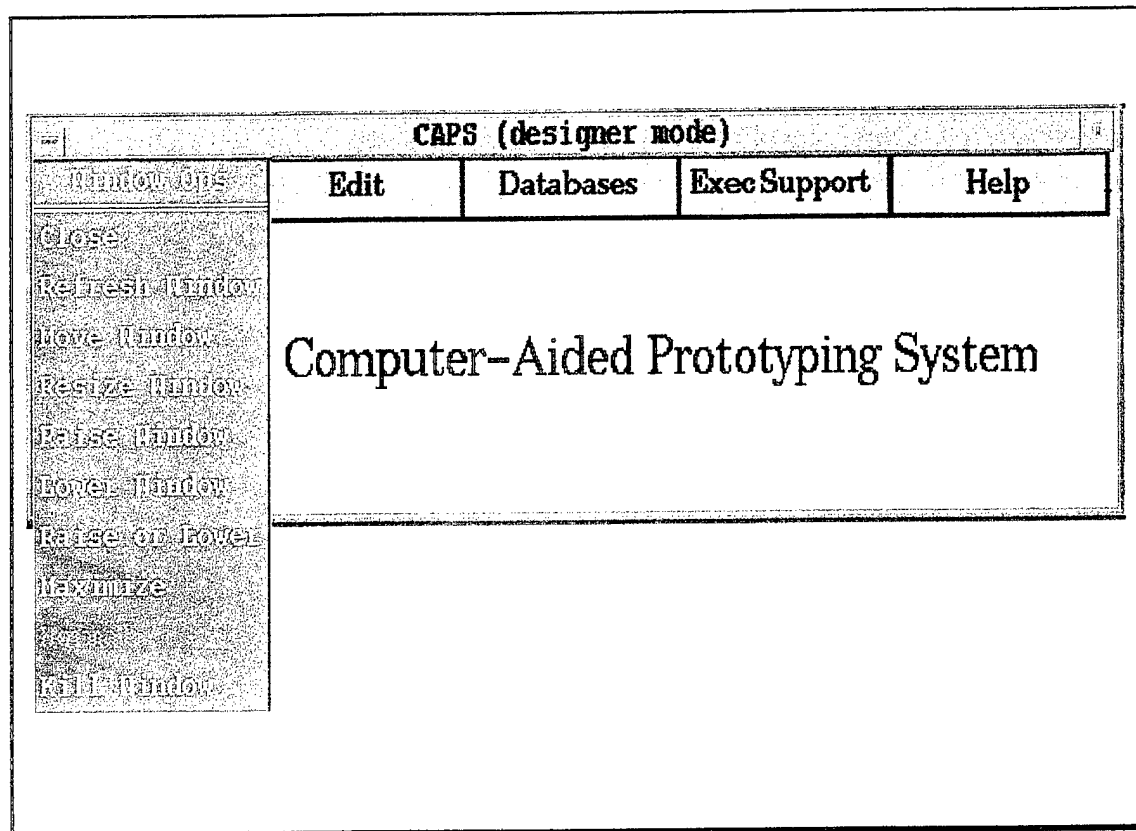
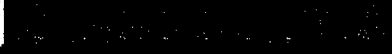










Figure 3.14. Unix workstation window menu pull-down menu.



<b>Table 3.2. Window Menu Option Buttons and Functions</b>	
<b><i>Option</i></b>	<b><i>Function</i></b>
	<i>Close the current active window.</i>
	<i>Refresh the current active window.</i>
	<i>Use the keyboard to move the window to another position.</i>
	<i>Use the corners of the window to change the size of the window.</i>
	<i>Bring the current active window to the front of the screen.</i>
	<i>Send current active window to the back of the screen.</i>
	<i>Enlarge the window to its maximum size.</i>
	<i>Dimmed when not available. Restores window to its normal size.</i>
	<i>Kills both the current active process and closes the current active window.</i>

## **F. THE GRAPHIC EDITOR DRAWING SPACE**

### **1. Description**

The graphic editor drawing space is the rectangular area used to hold drawn symbols and display the graphical representations of operators, terminators, and data streams. See Figure 3.15.

## **2. Usage within CAPS**

**NOTE:** In CAPS, terminators are referred to as operators but a different symbol is used to represent this operator.

CAPS utilizes the drawing space by serving as a container which holds drawn symbols and displaying the organized graphical symbols which represent operators, terminators, and data streams.

To place symbols in the drawing space, the drawing tools which are available from the tool sidebar are used. Procedures to use the drawing tools are discussed in Chapter VII.

## **3. User Interface Elements**

The drawing space has two scroll bars ( horizontal and vertical) associated with it. Each has user interface elements that allow movement and control of what is on the screen.

### ***a. Horizontal Scroll Bar***

The horizontal scroll bar is located to the right of the drawing space. It is used to move in either the up or down direction to view the drawn graph that was previously not in view. See Figure 3.15.

### ***b. Down Arrow***

The down arrow moves in a down direction to display previously out of sight data.

### ***c. Scroll Box***

The scroll box indicates your relative position in the data shown in the drawing space, and you can move the scroll box along the scroll bar to display a different part of the

drawn prototype.

*d. Up Arrow*

The up arrow moves in an up direction to display previously out of sight data

*e. Vertical Scroll Bar*

The vertical scroll bar is located across the bottom of the drawing space. It is used to move in either a left or right direction to view the drawn graph that was previously not in view. See Figure 3.15.

*f. Left Arrow*

This down arrow moves in the left direction to display previously out of sight data.

*g. Scroll Box*

The scroll box indicates your relative position in the data shown in the drawing space, and you can move the scroll box along the scroll bar to display a different part of the drawn prototype .

*h. Right Arrow*

The right arrow moves in the right direction to display previously out of sight data.

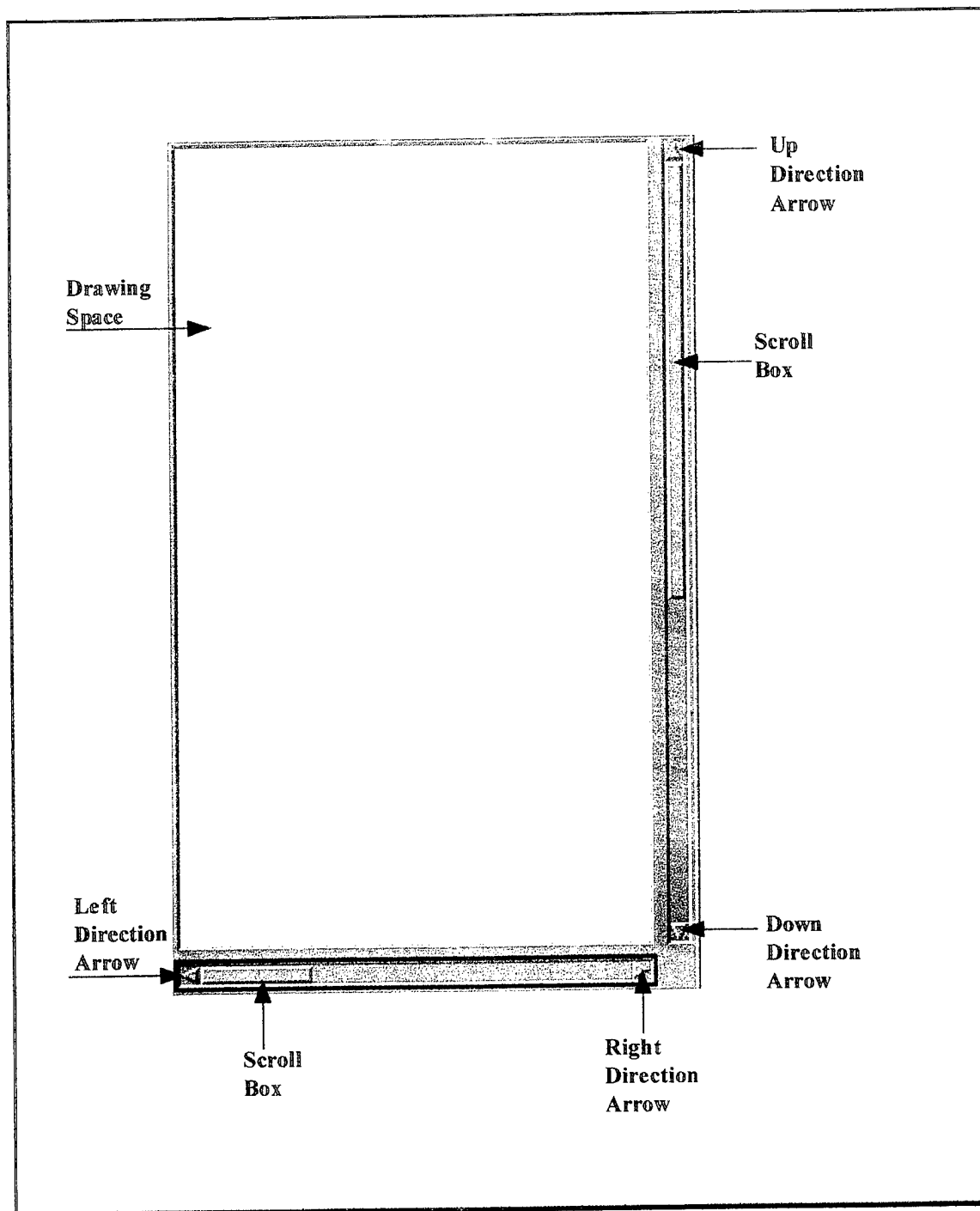


Figure 3.15. The graphic editor drawing space.

## **G. THE MENU BAR AND PULL-DOWN MENUS**

### **1. Description**

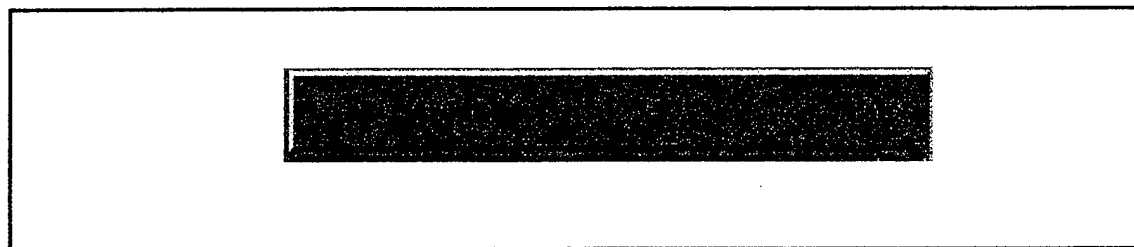
This chapter illustrates the Graphic Editor menu bar. The Graphic Editor menu bar is the horizontal row just below the title bar which displays the names of pull-down menus. The names of the pull-down menus are called menu items.

### **2. Usage within CAPS**

CAPS uses the menu bar to provide a menu-driven interface to allow the user to select items from a menu rather than type in a series of commands. An advantage of the menu bar is that all options are shown on the screen at the same time; with a menu there are no hidden commands for you to remember.

### **3. User Interface Elements**

The menu bar has four menu items (Graph, Draw Options, and Help) . Figure 3.16 illustrates the Graph Editor menu bar. Figure 3.17 illustrates the Graph pull-down menu. The table (Table 3.3) which follows lists the Graph menu options and functions.



**Figure 3.16. Graph Editor menu bar.**

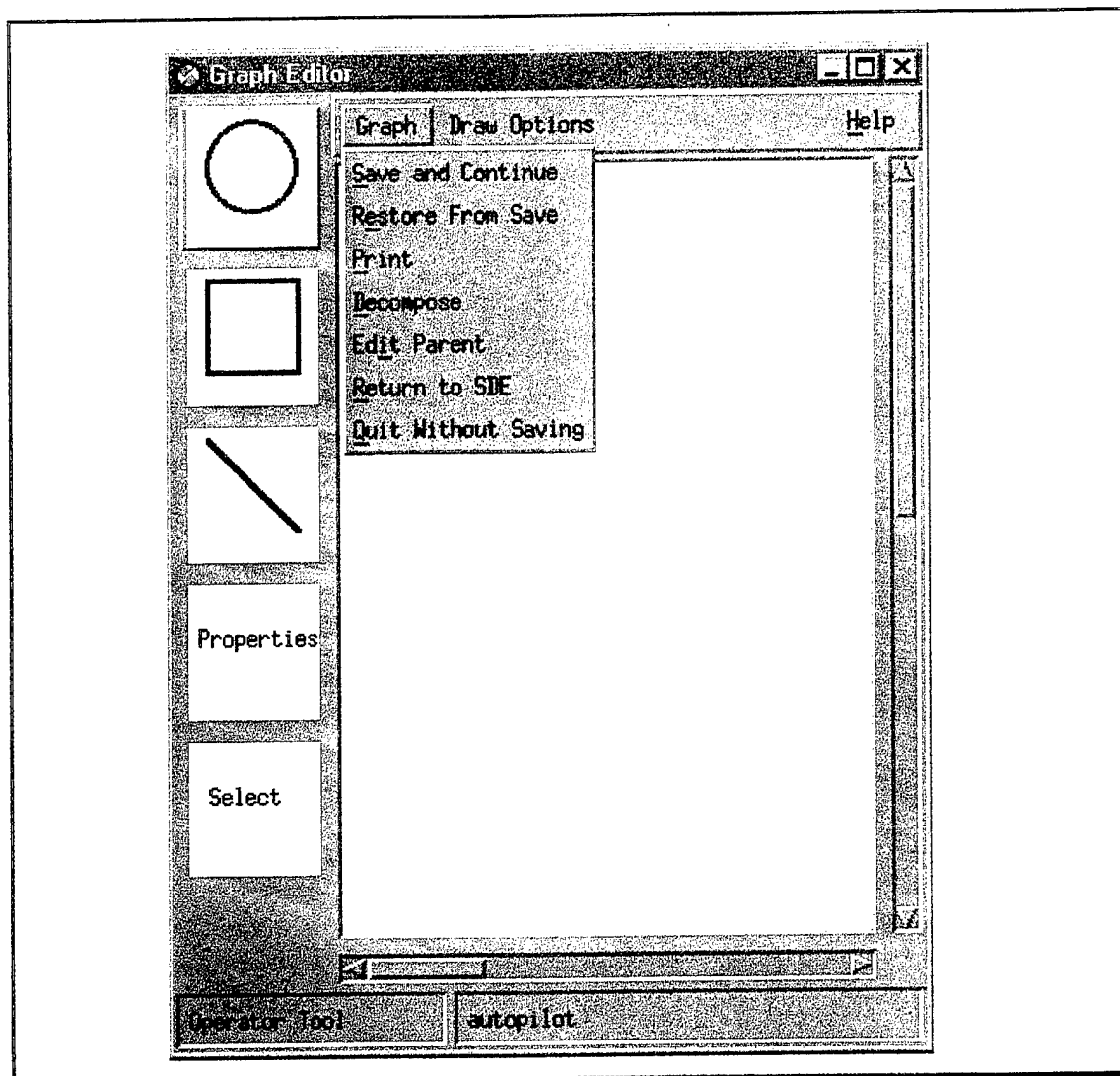


Figure 3.17. Graph pull-down menu.

**Table 3.3. Graph Menu Buttons and Functions**

<i>Menu Option</i>	<i>Function</i>
	<i>Saves the current PSDL graph in memory and control of the program remains within the Graphic Editor.</i>
	<i>Displays the contents of memory from the last save.</i>
	<i>Prints a screen dump of the viewable drawing space.</i>
	<i>Breaks down an atomic operator into a composite operator. This includes terminators which are known as operators in CAPS.</i>
	<i>Allows a user to edit the parent of a composite operator.</i>
	<i>Returns control of the program to the syntax-directed editor.</i>
	<i>Lets a user exit the Graphic Editor without saving the contents of the prototype file.</i>

The Draw Options (Color, Font, and Undelete Operator) is reflected below in Figure 3.18. Table 3.4 that appears on the next page lists the Draw Options menu options and functions.

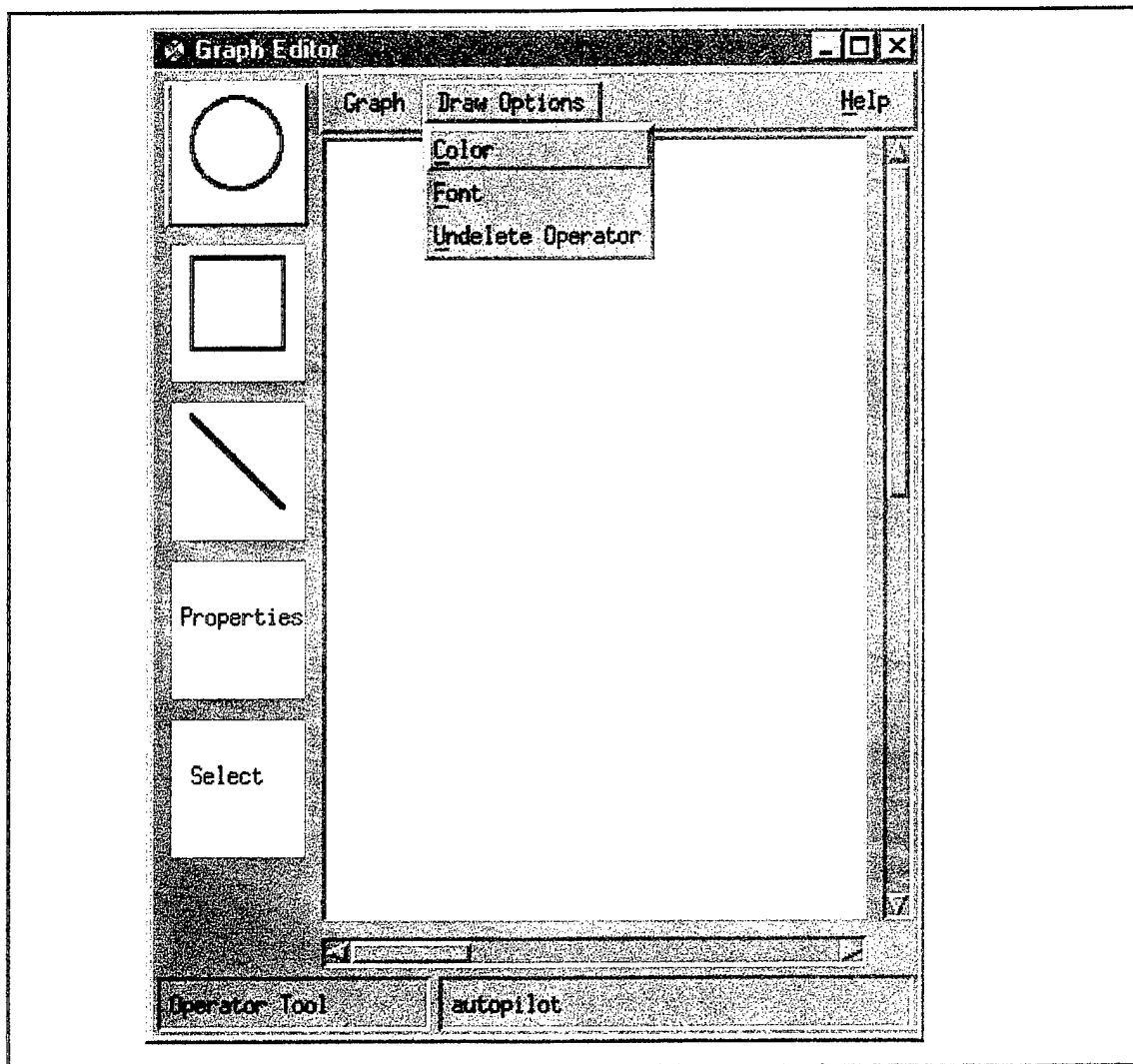


Figure 3.18. Draw Options pull-down menu.



**Table 3.4. Draw Options Menu Buttons and Functions.**

<b><i>Menu Option</i></b>	<b>Function</b>
	<i>Allows the user to change the color of items.</i>
	<i>Allows the user to change the font of the text.</i>
	<i>Allows the user to erase operators.</i>

The next illustration (Figure 3.19) shows the Help pull-down menu. Notice that this button is located on the menu for completeness. There is no on-line help linked to this help button. Table 3.5 lists the Help menu option and function information.

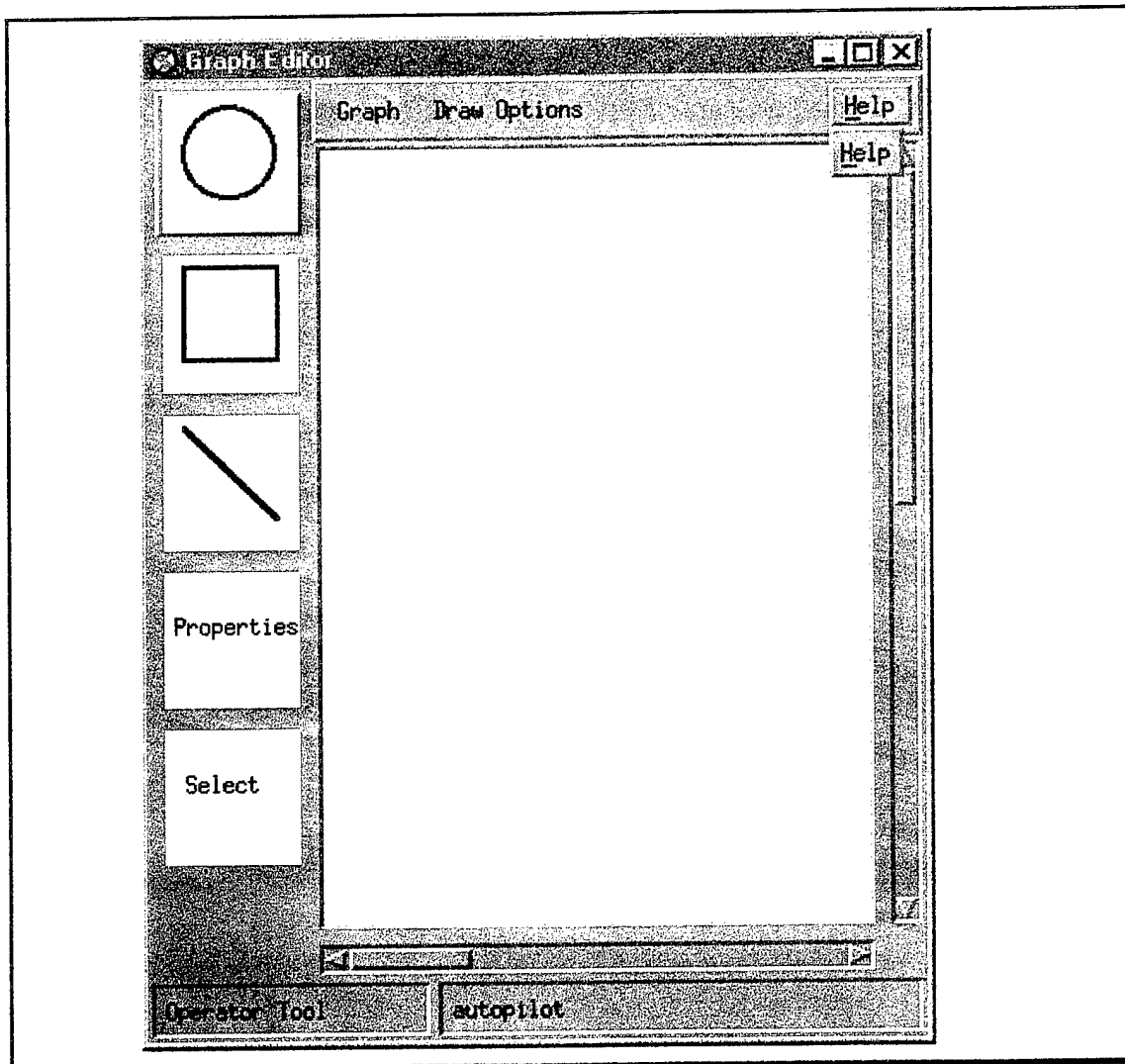


Figure 3.19. Help pull-down menu.

Table 3.5. Help Menu Button and Function.

Menu Option	Function
Help	Online help not implemented in Release 1.1.

## **H. THE TOOL SIDEBAR**

### **1. Description**

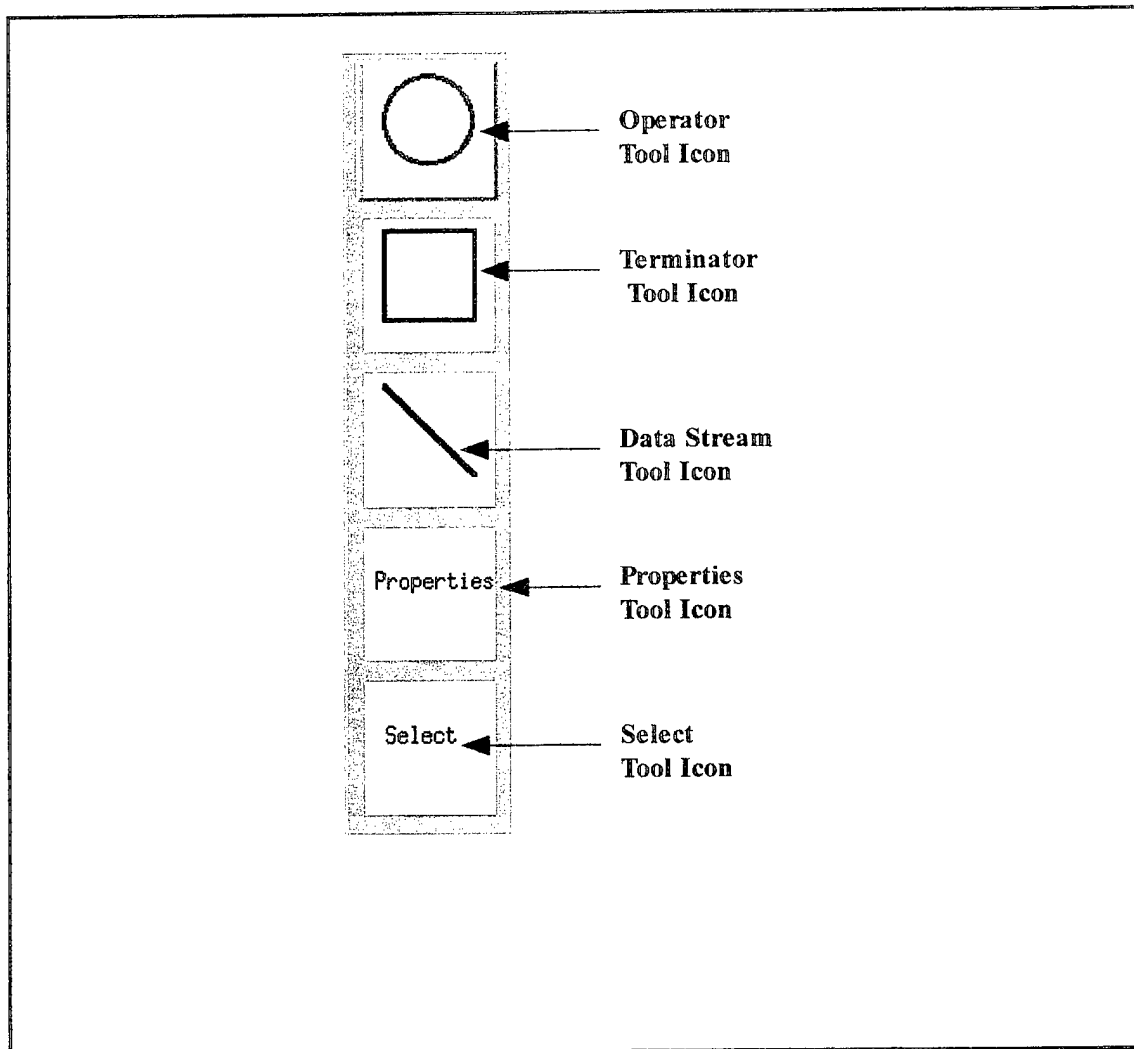
This Chapter explains the function of the tool sidebar and its usage within CAPS. Figure 3.20 shows all the buttons and icons which make up the tool sidebar; but labels only the icons to discuss at this time. These icons represent the symbols that are used to draw operators, terminators, and data streams.

### **2. Usage within CAPS**

This tool sidebar has many significant functions to name a few it uses tools to draw graphical representations of prototypes, uses dialog boxes to issue commands to add timing constraints attributes to the prototype, and issue commands to select prototype components to place in the drawing space or to select prototype components for editing.

### **3. User Interface Elements**

Figure 3.20 shows the labeling of the icons available from the tool sidebar.



**Figure 3.20. Graph editor drawing tool icons.**

#### **4. Drawing Tools and Icons**

To create a data flow diagram, choose a button from the choices of drawing tools and icons available from the tool sidebar shown in Figure 3.20. The drawing tools and icons have specific functions within CAPS. They are as following:

- To place what is called an operator in the drawing space, move the mouse pointer to the desired position in the drawing space and click the Operator Tool from the tool sidebar. You can continuously hold down the SELECT mouse button and move the mouse pointer to resize the drawn symbol.

As shown in Figure 3.20, the Operator Tool is the button with the circle as its icon.

The circle is the symbol used to represent an operator in a data flow diagram. In a graph, the circle represents a vertex. In a network, the circle represents a node. Operators are sometimes referred to as bubbles.

- To place an what is called a terminator in the drawing space, move the mouse pointer to the desired position in the drawing space and click the Terminator Tool from the tool sidebar. You can continuously hold down the SELECT mouse button and move the mouse pointer to resize the drawn symbol.

As shown Figure 3.20, the Terminator Tool is the button with the square as its icon.

In CAPS, since terminators are considered operators, the square or rectangular symbol is used to indicate that the object is external to the system. All other functionality is the same as an operator. The square or rectangular symbol represents a terminator in a data flow diagram. In a graph, the square or rectangular symbol represents a vertex which is external to the design. In a network, the square or rectangular object represents a node.

- To add what is called a data stream to the prototype design in the drawing space, position the mouse pointer on/over the Stream Tool. Click and release the SELECT mouse button. In the drawing space position the mouse pointer at a starting point; then move the mouse pointer to an ending point then click and release the SELECT mouse button.

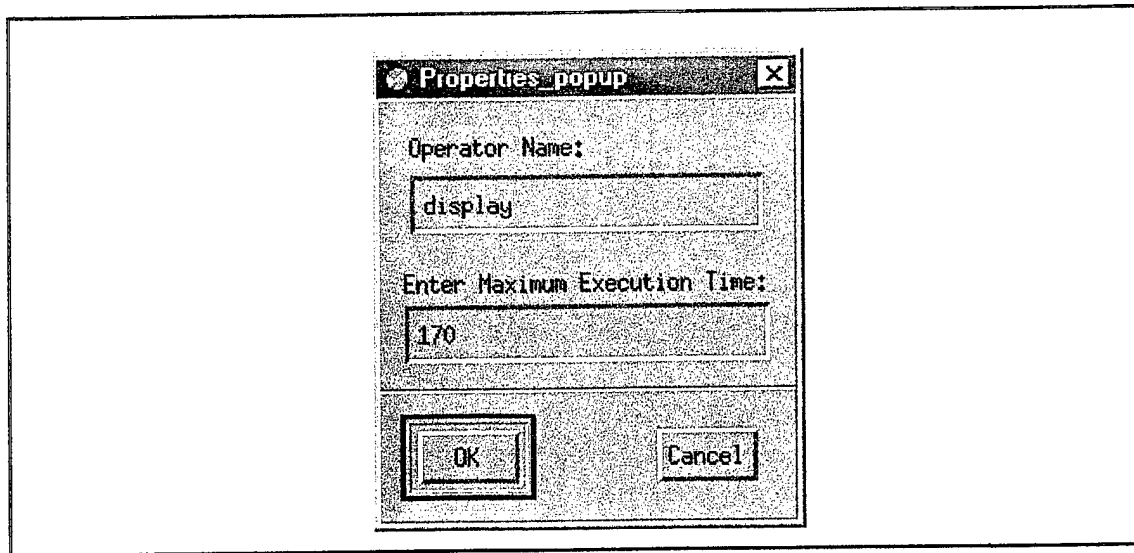
As shown in figure 3.20, the Stream Tool is the button with the line. line represents a stream in a data flow diagram. In a graph, the line represents an edge. In a network the line represents a communication link. These can be any point within a bubble. Curved streams are created by clicking on intermediate points between the starting and ending points.

External streams (inputs or outputs of composite bubbles) are represented as lines with unconnected ends. These are created by positioning the mouse pointer outside any bubble, or by double clicking to end the stream outside the bubble.

## **5. Properties Tool and Icon**

From Figure 3.20, the Properties Tool is labeled as Properties. In CAPS, messages, timing and control constraints are expressed in text. One tool used to add this information to a data flow diagram or PSDL graph is the Properties Tool.

To select the Properties tool, position the mouse pointer on/over its button. After clicking the Properties button, a Properties dialog box appears on the screen. Figure 3.21 shows the Properties popup dialog box.

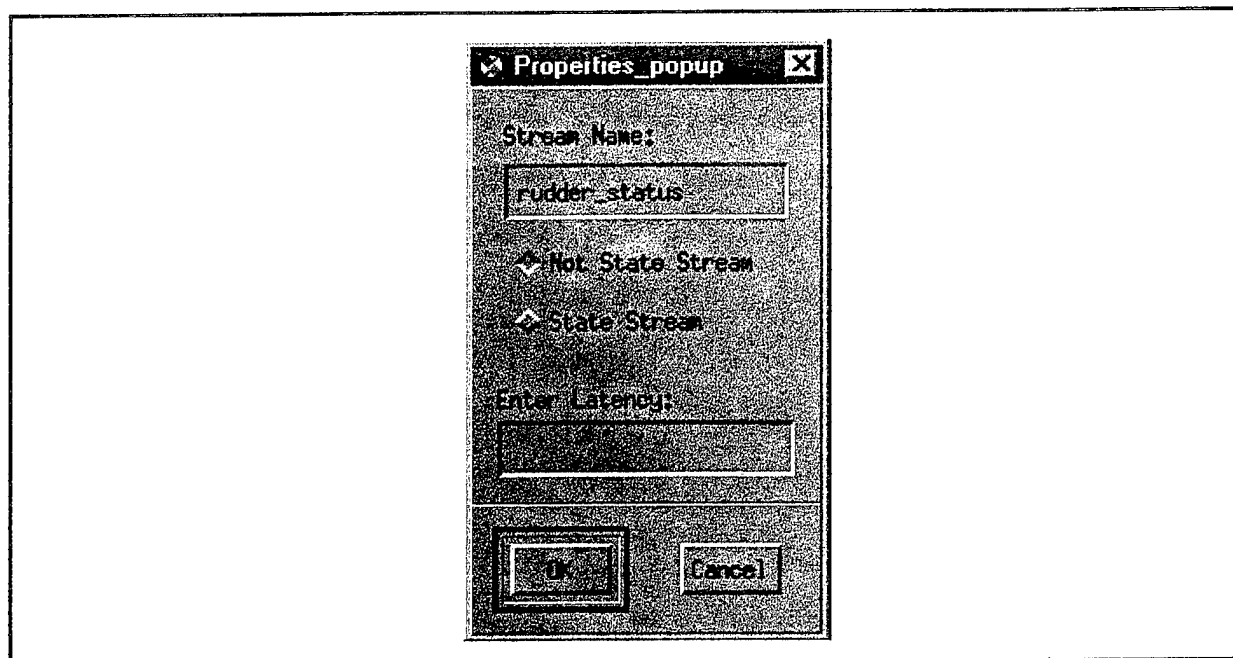


**Figure 3.21. Operator Properties popup dialog box.**

As shown in Figure 3.21, the operator properties dialog box provides the name of the operator selected, and provides a text box which allows the user to annotate the maximum execution time (MET) attribute. Once, attributes have been entered, press the Enter or Return key on the keyboard for the system to accept input.

#### **6. Stream Tool and Icon**

Figure 3.22 shows the data stream Properties popup dialog box.



**Figure 3.22. Stream Properties popup dialog box.**

As shown in Figure 3.22, the stream properties dialog box provides the name of the operator selected, allows user to indicate if stream is or is not a state stream, and provides a text box which allows the user to annotate the latency attribute. Once, attributes have been entered, press the Enter or Return key on the keyboard for the system to accept input.

## **7. Select Tool**

From Figure 3.20, the Select Tool is labeled as Select. In CAPS, designs are often edited, and rearranged. The tool to select which objects will be edited and rearranged is done with the Select tool. To select the Select tool, **position** the mouse pointer on/over its button and then proceed to make whatever changes you desire..





#### IV. CAPS DEVELOPMENT ENVIRONMENT GLOSSARY

The terms included in this chapter were selected from various computing dictionaries, books, journals, and the research papers reviewed for this thesis. Rather than list a source by each word, the references are included in the list of references. The terms listed in the following pages appear in bold text followed by its meaning in the CAPS development environment context.

**abstract data type**-In programming, information defined by the kind of data it can contain and the type of operations it can perform, rather than by the information it actually does contain or the operations it actually perform. For example, "date" is an abstract data type while "12/19/96" is actual data.

**access**-To obtain an open channel of communication with a software application or computer component so the user can work with it . For instance, a computer must access an attached modem before it can be used.

**access code**-An identifying string of characters used to regulate admittance to an application, data file, online service, local area network, etc. An access code distinguishes users who have legal access from those who do not. A password is an access code comprised of letters. See **password**.

**access mechanism**-A device or software driver that allows one component or application to communicate with another. For example, a device driver allows a software application to communicate with the computer system.

**access number**-The telephone number dialed by a modem that allows a computer to communicate with an online service or the Internet.

**access path**-The complete route, including the drive letter, directory, and all subdirectories, that directs an operating system to a file or application in a storage device. For example, the access path C:\WINDOWS\MyFiles which is an access path on a PC that runs Windows 95. On a Unix system the access path display the name of the machine (host) rather than a drive letter. For example se7> CAPS/ge/graph\_editor.

**access rights**-The claim a person or computer has to an open channel of communication with a software application or computer component. Access rights are often preserved by the use of access codes and passwords. For example, a user of a network might have access rights to use programs or files in one location, but not another.

**access server**-On a network, a computer designed to allow computers that are not part of a network to connect to and communicate with the network. The primary function of an access server is to provide a point of access for the connected computers. This function differs from that of a file server, although a single computer could serve as both. Although an access server allows other computers to access a network, a file server is the computer upon which network files are stored. The file server is accessible through the access server.

**access time**-The length of time required for a computer system to process a data requested and retrieve the data from memory, a storage device, or the Internet. This time may range from a few nanoseconds when accessing a file in the computer's memory to hours when retrieving data from the Internet.

**account**-In multiuser networks and operating systems CAPS environment, a registered connection primarily used for identification purposes.

**account policy**-In multiuser networks and operating systems, the set of rules that maintains order on the network and determines user access privileges.

**action statement**-Any distinct and executable command that initiates an action for the computer. The command may consist of a single word or a series of words and may perform one action or a number of related actions. An example is the print command, which in many software applications prints a file. An action statement differs from an expression, which designates a value instead of performing an action[Ref. 22].

**active**-Describes an application or file that is currently operational.

**active file**-Any data file that is currently open and able to receive or transmit data.

**active program**-Any application that is currently open and able to process data.

**active window**-In a graphical user interface, the display box currently affected by on screen cursor movement. The active window is usually a different color or shade of gray than other windows or is at the forefront of the screen. Active windows usually contain what the user is currently working on.

**Ada**-A high-level programming language developed in the late 1970s and early 1980s for proprietary use by the U.S. Department of Defense. Ada can handle almost any task, from simple accounting applications to complex missile guidance systems and has been a popular programming language within the Department of Defense since its creation. Ada was named after the early computer pioneer Augusta Ada Byron, Countess of Lovelace (1815-1852). She was instrumental in helping Charles Babbage conceptualize the analytical engine, a device that would have performed mathematical and logical operations and stored information.

**alert box**-In a graphical user interface, a box that displays a message to warn the user about potential or real system error.

**application**-An executable program capable of performing a specialized function other than system maintenance (which is performed by utilities). Also called software.

**application developer**-The individual or company that devises the idea for a program, specifies its functions, and designs the layout. The developer does not necessarily write the programming code.

**application file**-The portion of an application that makes a program operate as opposed to user-created data files that are manipulated by an application file.

**application icon**-A small picture or image that represents an application or utility in a graphical user interface such as X Windows or Microsoft Windows. Clicking or, in most cases double clicking on the icon will start the program.

**application processor**-A processor that **expedites** an application's functions by eliminating unnecessary tasks. An application processor is designed or reserved for use with a certain application.

**application programmer**-An individual who writes programming code for a software application. Programmers eliminate bugs from a program and correct programming errors, so they are responsible for an application's technical performance.

**application window**-The enclosed rectangular or square portion of a computer screen that displays a program. In operating systems such as Unix, Microsoft Windows, and the Macintosh OS, application windows can be made smaller or larger by the user. Windows can also be placed side by side or one atop another so that information can be dragged with the mouse from one window to the next.

**architecture**-The design of a computer system, hardware component, or operating system. Architecture refers to the entire structure and the information that makes it functional. Software applications are usually not considered a part of this system because they most often only perform a task and are not responsible for making a hardware component run.

**arrow keys**-On a keyboard, the keys imprinted with arrows, each indicating a specific direction, that often control on-screen cursor movement or select among menu choices.

**asterisk**-The name of the \* character on computer keyboards, usually engaged by pressing the Shift + 8 key combination. The asterisk represents multiplication in mathematical expressions and is often used as a "wildcard" variable for data file searches. For example, a search for the PSDL files \*.psd finds every file that ends with this extension in the designated directory or drive; a search for the file \*.\* would find every file in the designated directory or drive.

**atomic operator**-In the CAPS development environment, an operator that can be realized by an implementation stored in the software base or supplied by the software engineers.

## B

**background**-In multitasking environments such as Unix, Microsoft Windows, or Macintosh, several applications can run simultaneously. One runs in the foreground, while the others run in the background. The application or window in the foreground is active and can accept user input via a mouse, keyboard, or other device. Applications in the background cannot accept user input, but they can still run internal processes such as printing, reading or writing data to the hard drive, or performing calculations. In Windows, users can move background applications to the foreground by pressing the Alt+Tab key combination or by clicking on a background window.

Background can also refer to the color of the screen in the environment. Background colors can be selected according to the user's preference.

**Backspace key**-A key, usually located in the upper-right corner of the main section of the keyboard, that causes the cursor to move back one position. In word processing programs, the Backspace key often erases the character immediately preceding the cursor, while the Delete key erases the character following the cursor.

**blinking**-Disappearing and reappearing repeatedly. Cursors are familiar blinking objects on most computer screens.

**block cursor**-A type of cursor that appears as a small rectangle on the display screen, equal to the size of one character.

**border**-A line or other graphical element surrounding an area like a picture frame.

**box**-A graphical area that appears on-screen, usually with information or a request for user input, in a graphical user interface. Boxes are different from window in that they usually contain an entire application. In some instances, a box might also be a synonym for computer.

There are many different types of boxes. For example, **alert boxes** are boxes that suddenly appear on the screen to give the user information. **Dialog boxes** are boxes that request some type of information from the user. Boxes can also be small rectangular icons that control windows. **Zoom boxes**, for example, enable you to make a window larger or smaller.

**buffer**-A temporary storage area in a computer's memory, usually RAM (random access memory), that holds recent changes to files and other information to be written later to the hard drive. Because hard drives are relatively slow compared to RAM, buffers speed up performance. However, buffers generally are wiped clean by power outages; saving a file moves the information to the hard drive. Print buffers allow printing in the background while the user moves on to another application or document. Buffers are also used by some transmission protocols. Incoming data might be stored in a buffer until it is verified.

**buffering**-The act of saving information to a buffer, then passing it on.

**button**-An area on a computer screen in a graphical user interface that performs some function when selected by the mouse or, in some cases, by pressing the Enter or Return key. On-screen buttons look some like their real-world namesakes. Usually they include labels or some graphical element to denote their function.

In a graphical user interface, a dialog box element that lets the user select an option or command. In addition to application-specific buttons, almost all dialog boxes contain a **Cancel** button that allows the user to abort the current operation, as well as the **Ok** button, used to confirm a selection. See **click**, **dialog box**, **mouse**.

## C

**C**-A high-level programming language developed in the 1970s. C is a compiled language containing only a small number of built-in, machine-dependent functions. The majority of functions are machine-dependent, which means they are linked to a particular computer. Machine-dependent functions can be used on different computers with little or no change.

**C++**-An object-oriented version of C, developed in the early 1980s. This version has all the capabilities of C, as well as object-oriented programming capabilities. This language is popular for programming graphical applications.

**Cancel**-A button in almost any dialog box in a graphical user interface that allows a user to exit the dialog box without making any specified changes. It returns any settings to their status before the box was opened.

**capital letters**-Upper case letters, such as *A, B, C*, and so forth, as opposed to the lowercase *a, b, c*. Also called capitals or caps.

**caret (^)**-The symbol usually found on the number 6 key at the top of the main part of a computer keyboard. Some programming languages use the caret as an exponential operator. For example, the phrase  $4^2$  means the number 4 to the second power, or 4 squared. In computer instructions, the caret is often used to represent the Ctrl key on the keyboard.

**carriage return (CR)**-The control character that instructs a printer or computer to go back to the beginning of the current line of text. Unlike the Return or Enter key, it does not advance automatically to the next line. To advance to the beginning of the following line, the carriage return is combined with the linefeed character; this combination is called carriage-return/linefeed (cr/lf).

**cascading menu**-A secondary menu that appears next to the original menu when an option with a menu of its own is selected. This arrangement is common when a menu item has many related commands. Some cascading menu items have their own cascading menus in turn; this multilayer menu system is where the term cascading came from, as each menu flows from the one before it.

**channel**-The path taken by information exchanged between two places on the inside or outside of a computer.

**character**-One byte of information that represents a single symbol.

**character-based interface**-A nongraphical interface, using only text for interaction between the user and the computer. In this type of interface, users type commands instead of using graphical elements (icons) to give instructions to the computer.

**character style**-Any attribute applied to a character, including **bold**, *italic*, underline, and ALL CAPITALS.

**child**-A process started by another process, called the parent. It also can be a data record derived from another data record (again, the parent). For example, if the parent record is a student's academic record, a child record might be the student's grades in a particular class.

**choose**-In a graphical user interface, to select an option or command.

**circle**-In the CAPS development environment, a symbol which represents an operator in a PSDL graph or bubble in a data flow diagram or a node in a network.

**click**-Using a mouse by pressing and releasing its button one time without moving the mouse. This action is used to select objects in a graphical user interface pressing instruction might say "Click the OK button." See **click -and-drag**, **drag**, and **double click**.

**close**-To exit a file or program. In text base operating systems this action often called exiting or escaping, but in graphical user interfaces such as X Windows, close is sometimes used when the operator exits by closing the window in which the program is running. Closing a window removes it from the screen and ends the program.

**closed file**-A file not currently being used by an application.

**command**-An instruction to a computer program. Commands can be typed at prompts or chosen from menus in graphical user interfaces.

**command-based interface**-See command-line interface.

**command button**-In a graphical user interface, a rectangular box containing a command. When a user clicks the command button, the computer carries out the command.

**command-driven system**-A system in which the user types commands using the keyboard instead of choosing from a list of options, as in a menu-driven system.

**command-line**-The line on which the user types commands to be carried out by a program. This is the feature of a text based interface such as MS-DOS, as opposed to a graphical user interface such as X Windows or Microsoft Windows.

**command prompt**-See **prompt**.

**compiler**-A computer's interpreter. A program that changes the high-level source code of a programming language such as C into the basic machine language a computer understands. High-level programming languages come with compilers.

**computer**-A machine that accepts input, processes it according to specified rules, and produces output. The ENIAC, a massive calculator built during World War II that is considered by many to be the first electronic computer, and other early computers were based upon vacuum tubes, and later that decade, integrated circuits replaced the individual transistors. By the mid-1960's transistors.



**computer-aided engineering**-The use of software to test engineering designs created on computers. Also can refer to applications that analyze the designs.

**computer-aided software engineering (CASE)**-The use of computers in software development. The term also can refer to software used in program development.

**computer-aided testing (CAT)**-The use of the computer to test, designs, particularly those created using computer-aided design programs.

**context-sensitive help**-A feature in a program that gives the user information about the current command or operation without requiring the user to leave the program.

**composite operator**-In the CAPS development environment, an operator that can be decomposed into a network of more primitive operators presented as enhanced dataflow diagrams.

**cursor**-A marker on-screen that shows where current input or output is going to happen. It may appear as a blinking vertical line, solid or blinking box, an underline, or a caret.

**cursor position**-The location of the cursor on the screen. In text mode, this location is represented by a line number and character number and will be where the next typed character appears.

## D

**dataflow**-In the CAPS development environment, a First In First Out (FIFO) buffer of capacity one that connects synchronized operators. Dataflow is also the data in a data stream that represents discrete transactions, and is removed from the stream when it is read.

**data stream**-In the CAPS environment, a data stream is a pipeline through which data of known composition flows. This pipeline transmits data values from one operator to another. In a computational graph or PSDL graph, a data stream is known as a directed edge.

**debugger**-A program that locates the errors (bugs) in a program by proceeding slowly through the program, carefully checking the data structures and the program logic. A debugger verifies and validates a program for proper execution, placement of keywords, system and application parameters, and variable content. Some debuggers are relatively simple; others are more complex and may check spelling, performance, duplication, and unused or undeclared variables. Debuggers are tailored to a specific authoring language and are most often a part of an authoring system. In the CAPS development environment, the debugger monitors timing constraints and various aspects of design integrity as the prototype

runs, reports failures, and lets the designer adjust deadlines.

**deadline**-In the CAPS environment, the deadline is the maximum duration time from the triggering of the operator to the completion of its operation.

**default**-The standard setting predetermined by your computer, that is engaged when the user fails to denote a specific alternative. Defaults are generally the most often used settings for a particular program.

**default button**-In a graphical user interface, a button that has been predetermined to be the most often used option or command. For example, in Windows, the default button is often the Ok button.

**delete**-A command used to remove data from a storage medium.

**deselect**-The process of removing the highlighting from one or more choice or options. See also **select**.

**dialog box**-In a graphical user interface, an on-screen text box that provides users with information and explains possible options. After the user has clicked on the interests, it would present another set of options, such as list of formatting styles. Dialog boxes filled with options would continue to be placed on top of each other until the file or document is created. A dialog box opens when more information is needed from the user before the program can continue. A dialog box may contain several different elements, including text boxes, list boxes, command button, and drop-down list boxes, depending on the purpose of the dialog box, but it does not have to contain all these elements at the same time.

**dimmed command**-A command in a graphical user interface that is gray or shaded, rather than distinct and vibrant, to indicate it is unavailable for use. Dimmed commands cannot be use until some other operation has been performed.

**direct manipulation**-In a graphical user interface, the process of working with objects themselves using a mouse or other pointing device, rather than using menu selections to manipulate the objects. Using drag-and-drop to print a file, or using the mouse to adjust the size of a window are both examples of direct manipulation.

**directory tree**-In a graphical user interface, a visual representation of the branching structure of all the directories, subdirectories, and files on a disk. Directories, subdirectories, and files may be shown by name or represented by icons.

**disable**-To turn a function off or prevent something from happening. In a graphical user

interface, disabled menu commands are often shown in gray to indicate that they are not available. See **dimmed command**.

**double click**-To press and release the mouse button rapidly, twice in quick succession, without moving the mouse. Double clicking is used to select an object as well as to initiate an action. For example, if you double-click a program icon, you select that program, and also start the application running. See **click**.

**drag**-In a graphical user interface, to move a selected object using the mouse. You place the mouse cursor on the selected object, hold down the mouse button while moving the mouse to the new location, and release the mouse button; and the object is inserted.

**draw tool**-In a drawing or illustration program, the command that turns the cursor into a pen-like tool used to create object-oriented graphical images. Draw tools often allow the user to manipulate lines, circles, curves, polylines, and rectangles.

**drop-down list box**-In a graphical user interface, a dialog box element that helps the user choose one item from a list of possible alternative. Drown-down list boxes are often used when there is not enough space in the dialog box to use a list box.

**drop-shadow**-In a graphical user interface, a shadow shown below and to the right of a dialog box, giving the impression that the dialog box is three-dimensional and is raised from the screen slightly. Drop shadows are also used in printing.

**dynamic scheduler**-In the CAPS development environment, the dynamic scheduler allocates time slots for operators that are not time critical.

## E

**edit**-To make any change to the contents of a file. Many programs, such as spreadsheets, have special edit mode, designed to make the editing process easier. In other programs, like word processors, you use the same screen and program functions to edit a document that you used to create it.

**editor**-Also known as a text editor, a type of program that let you create and manipulate text files. There are several kinds of editors. A regular, full-screen by navigating the file with the cursor or arrow keys. A line editor is a little awkward, numbering each line in the document. To make a change, you first must indicate to the editor the number of the line you want to change so it will take you there. An editor is not as full-featured as a word processor; generally text editors do not include features such as italics or spell checking.

**ellipsis**-In a graphical user interface, an ellipsis(...) next to a menu selection indicates that you will be asked for additional information before the command can be performed. When you select the command, a dialog box opens to obtain this extra information. See **dialog box**.

**enable**-to turn a function on or allow something to happen. When a function is enabled, it is available. In a graphical user interface, enabled menu commands are often shown in black type to indicate that they are available. See **disable**.

**Enter key**-The key on the keyboard used to tell the operating system or application program that input is complete. Also known as the Return key. See **Return key**.

**environment**-An area of memory used to store system-wide information, such as the path and details of the command prompt. Users or application programs place values in the environment by using the SET command. The complete set of hardware and software resources made available to any user of a system, including the processor used, the operating system, system utilities, and, very often, the user interface. The operating system that a program needs in order to execute.

**event-driven**-Any program designed to react to a keystroke or a mouse click, rather than forcing a user to go through traditional menu selections and on-screen prompts.

**evolution control system**-One of the tools associated with the project control system in the CAPS development environment. Its designed to give automated help to the task of coordinating concurrent efforts of prototype design teams and managing the multiple design versions that are produced.

**execution support system**-One of CAPS subsystems composed of a translator, static scheduler, dynamic scheduler, compiler and debugger.

**Exit**-A command user to return control from a secondary command processor to the parent processor, if one exists, otherwise **Exit** has no effect. See also **command**.

## F

**feature**-A desirable or otherwise notable aspect of an application or piece of hardware.

**file**-Information stored as a series of bits organized in away that can be recognized by computer software. Not all files can be used by all software. Files can be parts of a program or used by data programs. Files are recognized by special names.

**foreground**-The foremost task or object on a computer system, in an application, or on a

network. Tasks in the foreground are those the user is working on at the moment, which usually have access to more microprocessor time. Other tasks, those in the background, run more slowly and receive less microprocessor time. In windowing operating system, the window on "top" of the other windows is said to be in the foreground. The terms also are used in some graphics applications, when some elements of a picture or layout lay on top of other elements.

**full-screen**-Occupying or being displayed across an entire computer screen. Full-screen video, for instance is displayed on the entire screen rather than being contained in a small window. Full-screen video or other graphics may have to be enlarged to fit a whole screen, which can result in a lower-quality picture.

**full-text search**-A search that examines the entire text of files being searched for certain keywords or phrases. More limited searches might only look at titles, headers, authors, or other information.

## G

**grabber**-In some applications, a special tool or cursor that enables you to grab objects on the screen and move them or manipulate them in some other way. A grabber cursor is often represented by a hand icon. See **cursor**.

**Graphic Editor**-A CAPS tool which uses the prototyping system description language (PSDL) to allow users to sketch graphical representations of a system using computational graphs. In the CAPS development environment, the vertices are known as operators, directed edges are data streams and the data streams allows the designers to refine the design by adding timing and control constraints in text form. The graphic editor's prototype design representations consist of non-procedural control constraints as a part of the specification of a hierarchically structured prototype, resulting in a preliminary, top-level design free from programming details.

**graphical user interface (GUI)**-A program interface that takes advantage of the computer's graphics capabilities to make the program easier to use. Well-designed graphical user interfaces can free the user from learning complex command languages. On the other hand, many users find that they work more effectively with a command-driven interface. Especially if they already know the command language.

**Graphics mode**-A computer display mode where an image is produced by pixels that create the image. Each pixel is a unit that can be manipulated.

## H

**handles**-When you select a graphic within a graphics program, a box ordinarily appears around it. Making up this box are small squares (usually four, six, eight), called handles. You can drag the handle boxes to reshape or resize the image.

**help**-Also known as online help. Information stored on disk that the user can access directly which provided assistance and advice on running the program. You can access this information from the Help menu in an application or by clicking on the Help button found in almost all dialog boxes. Help information may be general in nature or context sensitive, and specific to the task at hand. While most online help is not usually as complete as the information contained in a manual, it is useful if you are an occasional user of a program, or if you want to see an explanation of little-used program features.

**Help button**-In a graphical user interface, a button found in almost all dialog boxes along with the Ok button and the Cancel button. Use the help button to display online help information.

**Help index** - A choice in the Help menu that opens an alphabetic listing of all the help topics available. You can make a selection from this list, and view the topic you are interested in.

**highlight**-Any indication that a text block or an object has been selected with a mouse click or the arrow keys on the keyboard. The highlight may appear as a change in the color of text or a border appearing around an object is usually ready for some type of operation.

**horizontal scrolling**-The ability to move a document or spreadsheet beyond the limits of a screen to the left or right. Scrolling is usually performed by clicking an arrow on the top or bottom of a window.

## I

**I-beam cursor**-In a graphical user interface, a mouse pointer used when editing text. The I-beam cursor is thin enough that you can position it between text characters very for inserting text or to highlight text for editing.

**icon**-In a graphical user interface, a small screen image representing a specific element that the user can manipulate in some way. You select the icon by clicking a mouse or other pointing device. An icon may represent an application program, a document, embedded and linked objects, a hard disk drive, or several programs collected together in a group icon. See also **graphical user interface** , **group icon**, **group window**.

**iconic interface**-An interface composed of graphical images that represent the program or function to which they are connected.

**inactive window**-In an operating system or application program capable of displaying multiple windows on the screen, all open windows except the currently active window, the window that contains the cursor. If you click on an inactive window with the mouse, it becomes the active window. See also **active windows**, **cascading windows**, **graphical user interface**.

**insertion point**-The point at which text will be inserted when you type the next character. In a graphical user interface, this point is often indicated by an i-beam cursor, a blinking vertical line.

**integrated development environment** - Abbreviated IDE. A complete set of program-development tools, all run from single command user interface. An IDE may contain a text editor, compiler, debugger, and disassembler, as well a profiler for program performance analysis.

**interface standard** - That point where a connection is made between two different parts of a system, such as between two different parts of a system, such as between two hardware devices, between a user and a program or operating system, or between two application programs. In hardware, an interface describes the logical and physical connections used, as in RS-232-C, and is often considered to be synonymous with the term "port." A user interface consists of all the means by which a program communicates with the user, including a command line, menus, dialog boxes, online help systems, and so on. User interfaces can be classified as character-based, menu-driven, or graphical. Software interface are application program interface; the codes and messages used by programs to communicate behind the scenes. See also **social interface**.

**integrate**-As a verb, integrate refers to the action of two or more hardware or software components working together as a system. As an adjective, it refers to a single unit that is a conglomeration of individual components.

**integrated software**-Software consisting of several applications specifically designed to work together. For example, one package might include a word processor, a spreadsheet, and a database. The appeal of integrated software is that it makes it easy for a recreational computer user to transfer data among the included applications. But the programs in an integrated software package usually do not offer the high-end capabilities of individual, standalone applications.

**interactive**-A quality of a computing device that requires input from a user. Most programs are interactive. Non-interactive programs can run to completion without any human input.

**interactive program**-A communication link in a computer between hardware and software components. A user interface is a way a user communicates with a computer.

## K

**key combination**-Also known as a shortcut keystroke. In menu-driven and graphical user interface, certain menu command respond to keystroke combinations used directly from the keyboard, rather than the more usual method of opening menu and using the mouse to choose the command.

**key-in**-To put information into a computer using a keyboard or numeric pad. Often use to describe the inputting of large amounts of data into a database.

**keystroke**-The action of pressing and the releasing a key on the keyboard to initiate some action or enter a character.

## L

**label**-A word or group of words that identifies something. A label, in the physical sense, is a sticker, piece of paper, or some type of tag affixed to an object to identify it.

**latency**-In the CAPS development environment, the upper bound on the time between the instant a data value is written into a stream and the instant that data value can be read from the screen.

**launch**-To start an application program running, usually by double-clicking on its icon with the mouse.

**list box**-In a graphical user interface, a dialog box element that helps the user make on choice from a list of possible alternatives. You make a selection from a list box clicking on your choice with the mouse, just as you would select from menu. If there are too many items to fit into the list box at the same time, a scroll bar appears. List boxes are often used to select a file or document name.

**login** - Also know as logon. To establish a connection to a computer system or online service before using it. Many system require the entry of an identification number or a password before the system can be accessed. **See also logout.**



**logout** - Also known as logoff. To relinquish a session and sign off a computer system by sending a terminating message. The computer may respond with its own message, indicating the resources consumed during the session, or the period between login and logout. Logging out is not the same as shutting down or turning off the computer. See **login**.

## M

**maximize execution time**-In the CAPS development environment, the maximum amount of CPU time required to execute an operator under worst-case conditions. The maximum execution time is considered to be the longest time between the instant an operator begins execution and the instant it completes execution. In addition, the maximum execution is a time constraint, applied to each time-critical operator, representing the maximum time the operator may need to complete execution after it is fired, given access to all required resources.

**maximum response time**-In the CAPS development environment, the maximum duration allowed from the triggering of the sporadic operator to the completion of its operation.

**menu** - A list of the commands or options available in the program displayed on the screen. You select a menu item by typing a letter or number corresponding to the item, by clicking it with the mouse, or by highlighting it and pressing the Enter key. See also **menu-driven**, **pull-down menu**.

**menu bar** - A row of pull-down menu names, usually displayed in a line across the top of the screen or window, just below the title bar.

**menu-driven** - Describes a program controlled by selecting items from a menu, rather than typing a series of commands. One of the many advantages of menu-driven software over command-driven software is that all the options are shown on the screen at the same time; there are no hidden commands for you to remember. See also **user interface**.

**merger**-A part of the project control system, in the CAPS development environment, the merger helps to combine the product of two or more independently developed prototype changes, thereby facilitating parallel enhancements and applying common changes to multiple versions.

**minimize** - In a graphical user interface, to reduce the active window to an icon. To minimize a window, click on the button pointing downwards in the top right corner of the window, or use the Minimize command from the Control menu. The window becomes an icon at the bottom of the desktop where it occupies only a small amount of space, but is ready to be opened up at a moment's notice. After a window has been minimized, double-click on the icon to return the window to its original size. See also **maximize**.

**minimum calling period**-In the CAPS development environment, the minimum calling period is the smallest interval allowed between two successive triggerings of a sporadic operator.

**minimum period**-In the CAPS development environment, the minimum period is the lower bound on the time between two successive write events on the stream.

**mouse** - A small input device with one or more buttons used as for pointing or drawing. As you move the mouse in any direction, an on screen mouse cursor follows the mouse movements; all movement are relative. Once the mouse pointer is in the correct position on the screen, you can press one of the mouse buttons to initiate an action or operation; different user interfaces and file programs interpret mouse clicks in different ways.

A mouse has been standard equipment on the Macintosh family of computers for a long time, and with the rising popularity of graphical user interface such as Microsoft Windows on DOS computers, the user of a mouse is growing there, too.

You can connect a mouse to the computer in one of several different ways;

- A bus mouse requires a separate expansion board in the computer
- A serial mouse plugs into an unused serial port.
- A regular mouse plugs into the mouse port as on the Macintosh and IBM PS/2 computers.
- A wireless mouse is also available.

## N

**non-time-critical operator**-In the CAPS development environment, an operator which has no timing constraints associated with it.

## O

**OK button**-In a graphical user interface, a dialog-box element you can select to confirm an operation or move on to the next level in the program. Almost all dialog boxes contain an OK button. See also **button**.

**operator**-In the CAPS development environment, a state machine with zero or more state variables. Also a state machine whose internal states are modeled by state variables.

**operating system (OS)**-Software that controls a computer and its peripherals. Early operating systems, such as DOS and Unix, left a great deal of the operation to the user, but current Oses, such as OS/2 and Windows 95, handle many of a computer's basic functions.

**option button** - Also known as a radio button. In a graphical user interface, a small round button used to make an exclusive choice in a dialog box where only one option can be in effect at a time, like baud rate, or to choose between an ascending or a descending sort, for example. Only one option button can be selected at a time. If you click on the button to select it or turn it on, the button will be filled; you can click on it a second time to turn it off.

**option key** - A key on the Macintosh keyboard that, when pressed in combination with another key, produces special characters like boxes, international characters, and special punctuation marks.

## **P**

**path**-The complete description of the location of a file or directory in the file system. The path consist of all the directory names that must be accessed in order to get to a specific file.

**period**-In the CAPS development environment, the time interval between triggering times for the operator.

**periodic operator**-In the CAPS development environment, a time-critical operator that is activated by a temporal event. This operator is triggered by a data stream or regular timing constraint. The timing constraints are called periodic timing constraints.

**pointer**: A symbol that appears on the display screen and that you move to select objects and commands. Usually, the pointer appears as a small angled arrow. Text-processing applications, however, use an I-beam pointer that is shaped like a capital I.

**pointing device**: A device, such as a mouse or trackball, that enables you to select objects on the display screen.

**pop-up menu**-A menu displayed next to the element with which it is associated. A pop-up menu is usually only displayed on request; in other words, it is only displayed when you specifically ask for it.

**prompt**-The symbol at which commands are typed by the user to make the system perform functions in a command line interface. it is also called the command prompt.

**Print**-To transmit data such as text and graphics from a computer to a printer in order to create a hard copy of the data.

**prototype**-In the CAPS development environment, a prototype is an executable pilot version of the intended system that accurately reflects chosen system attributes.

**prototyping** - The creation of a trial or demonstration system, so that potential users can evaluate, refine, and comment on the design before work on the actual product begins. Tools for both hardware and software prototyping are available; indeed, some software prototyping tools can actually be used to generate the final code once, for example, a screen layout has been approved.

**prototyping system description language (PSDL)**-A high-level prototyping designed specifically to support conceptual modeling of real-time embedded systems, including the timing aspects of hard real-time systems in single and multi-processor hardware configuration.

**psdl editor**-See **syntax-directed editor**.

**PSDL graph**-In the CAPS development environment, augmented computational graphs created or modified in the Graphic Editor tool by selecting and arranging graphical symbols (circles, straight or curve lines, and square or rectangle box).

**pull-down menu** - A vertical menu that you pull-down from a set of menu names arranged in a menu bar across the top of the screen or the top of the window. To make a selection from a pull-down menu, you click on the item with the mouse, or use the cursor-movement keys to position the highlight over the item and press the Enter key, or type a special key combination.

## Q

**quit** - To exit the current application program in an orderly way, and return control to the operating system.

## R

**rapid prototyping**-An approach used in the CAPS development environment to quickly build and evaluate a series of prototypes.

**read** - To copy program or data files from a floppy or a hard disk onto computer memory, to run the program or process the data in some way. The computer may also read your commands and data input from the keyboard.

**real-time system** - Any computer system designed to respond immediately, as events occur in the real world. Computer systems used in airplane automatic pilots, patient-monitoring system, process control, or traffic control system are all real-time systems; computers that perform batch-processing operations are not.

**restore** - In a graphical user interface, to return a window to its original size after it has been maximized. To restore a window, click on the downward-pointing part of the double arrow in the top right corner of the window, or use the Restore command from the Control menu. See also **maximize**, **minimize**.

**reverse engineering** - The process of disassembling a hardware or software product from another company to find out how it works, with the intention of duplicating some or all of its functions in another product.

**revision control** - A program-development tool used to keep track of changes and revisions in program source code as software development continues.

**run** - To execute a program.

**run-time version** - A special, limited-capability release of software bundled with a single product, that allows that product to run, but not support any of the other applications capable of running in that same environment. In other words, the run-time version provides some but not all the features of the full product.

## S

**save** - To transfer information from the computer's memory to a more permanent storage medium such as a hard disk. As you work with your computer, you should save your work every few minutes. Otherwise, if you suffer a power failure or a severe program error, all your work will be lost because it is stored in memory, which is volatile, and when the power is removed, the contents of memory are lost.

**scroll** - In a graphical user interface, to move a window up, down, left, or right, in order to see information that was previously out of sight. See also **scroll bar**, **scroll box**.

**scroll bar** - In a graphical user interface, a vertical or horizontal bar at the right or across the bottom of a window that is too small to show all the necessary information at the same time.

At each end of the scroll bar, small arrow indicate the scrolling direction. To move through the document, click on the appropriate arrows. Within the bar, a small box called a scroll box indicates your relative position in the data shown in the window.

**scroll box** - In a graphical user interface, a small movable box located on one of the scroll bars. The scroll box indicates your relative position in the data shown in the window, and you can drag the scroll box along the scroll bar to display a different part of the document. This is usually faster than repeatedly clicking on the arrows at the ends of the scroll bars.

**select** - The act of choosing a menu item or highlighting an option. When you make a selection, you expect a specific action to result. **See also deselect.**

**slider** - In a graphical user interface, a control representing a quantity and the range of possible values within that quantity. Often used as a volume control or as controls in a color model.

**social interface** - A form of graphical user interface that employs images representing real objects rather than icon.

**software base**-In the CAPS development environment, the software base provides and keeps track of the reusable software components for realizing given PSDL specifications and Ada implementations.

**sporadic operator**-In the CAPS development environment, a time-critical operator that is activated by the arrival of dat. it is triggered by a data stream timing constraint that constraint is data.

**square**-In the CAPS environment, a symbol which represents a terminator in a PSDL graph. It also is known as a bubble, but it is a rectangular bubble indicating an external source. This symbol represents a node in a network.

**static scheduler**-Uses several algorithms to allocate time slots for operators with real-time constraints before execution begins. In the CAPS development environment, if this allocation succeeds, all the operators are guaranteed to meet their deadlines even in the worst case. If the static scheduler can not find a valid schedule, it provides diagnostic information about the cause of the problem and if it can be solved by adding more processors.

**syntax-directed editor**-One of the text editors in the CAPS development environment. The syntax-directed editor has the capability of verifying each program line entered to assure that it is syntactically correct without delaying entry.

## T

**Transportable Applications Environment Plus (TAE+)**-A windowing package developed at the National Aeronautics and Space Administration's Goddard Space Flight Center. TAE Plus provides either Ada or C code to create the user interface modules. In the CAPS development environment, TAE Plus was modified to meet the CAPS conventions.

**terminator**-In the CAPS development environment, a source or sink which represents an entity outside the context of the system that is a net transmitter or receiver of system data. In the PSDL graph, a terminator represents an edge.

**time-critical operator**-In the CAPS development environment, an operator which has at least one timing constraint associated with it.

**translator**-Generates code that binds the reusable components extracted from the software database. In the CAPS development environment, the translator main functions are to implement data streams, control constraints, and timers.

**toolbar**-A row of boxes, often at the top of an application window, which control various functions of the software. The boxes often contain images that correspond with the function they control. Toolbars can be turned on and off and often can be personalized with controls specific to an individual user's needs.

## U

**undelete**-To restore a deleted object or file using a utility program. Deleted files are not actually erased until the storage area where they reside is needed for another file. Therefore, it is sometimes possible to undelete a file.

**undo**-A command that reverses the action most recently completed. Some programs, however, let several commands be undone at once. Whenever a user chooses the undo button in a word processing program, the previous command is undone.

**Unix**-A powerful operating system developed by AT&T Bell laboratories in 1969 and used primarily by universities and mid-sized businesses. Written in the C programming language (which is popular for PC use because it takes less memory to use than other languages), this multiuser, multitasking operating system was designed for both large mainframes and minicomputers. It can be used on many platforms and can run a wider variety of hardware than other operating systems. This operating system is more popular for workstation computers on networks rather than individual PCs.

**user-account**-An individual set of parameters and profiles designed for one user of a multiuser system, such as for each family member using a home PC. It often consists of information about the user, such as the user's name and the password required to access the system. in addition in information about which files and programs the individual can use.

**user-friendly**-Any nonprofessional person (those who do not program computers or provide technical support for them) who works with PCs.

**user interface**-The part of the software the user works with, whether it be through commands or menus. Interfaces can be either text-driven, such as DOS, or graphical user interfaces (GUIs), such as Windows.

**user name**-A code used by individuals, in addition to a password, that let them access a network computer, an on-line service, or a bulletin board system.

## V

**version**-A number used to indicate a program's stage of development. A new software version, indicated by product name changes such as CAPS Release 1, indicates that the software has been updated or improved. Major upgrades are indicated when the number before the decimal point changes as in the previous example. Minor revisions are indicated by changes in the number after the decimal pint and small fixes, such as corrections of program flaws, are indicated by additional numbers or letters.

## W

**window**-A bordered area on-screen that contains an application or document, often used in graphical user interfaces (GUI). Each window can be enlarged, educed, or minimized to an icon, which temporarily removes it from view. A window provides for multitasking (meaning several programs can be run at once) and allow user to cut, copy, and paste information from one window to another. Additionally, windows can be arranged side-by-side (tiled) so that the user can see the full window for a document or application , or overlaid so that the frontmost window is seen in full while only the title bar shows for window in the background (cascade). Windows are enlarged or reduced by clicking on zoom buttons in the upper-right corner of a display. When an application is open , there is another zoom button (similar in appearance to the aforementioned ones) that resizes a document window within a program, allowing the user to see. Reduced windows can be resized further by placing the mouse pointer on their borders, pressing the left mouse button, and dragging to the desired position.

**workstation**-A setup composed of a computer and peripheral devices that enable some



someone to do their work. In terms of processing power, workstations fall between personal computers and minicomputers. Also can designate any computer connected to a network.

**write**-The transfer of data from a computer's memory to a storage or output device. The write process includes such functions as saving data to diskette and outputting a copy to a printer.

## **X**

**X-Windows**-A windowing and graphics system developed at the Massachusetts Institute of Technology for Unix workstations. In the X-Windows system, graphical user interfaces are created independent of computer hardware. Examples include Motif and OpenLook.

## **Z**

**zoom**-To make a window larger when using graphical interfaces. In many applications, a zoom box lets the user maximize a window, making it fill the entire screen. When the zoom box is selected again, the window returns to its original size.

## **V. FUTURE RESEARCH AND DEVELOPMENT**

The computer-aided prototyping software development environment is an ongoing software engineering research project at the Naval Postgraduate school. This project offers numerous opportunities for software engineers, DOD project managers, and graduate students to develop, test, and implement new research concepts to meet the current software engineering development challenges.

Due to the requirement to develop compatible software that can be executed on hard real-time embedded systems, it is essential that computer instructions are performed accurately and rapidly. In such an environment it is clear that a CAPS User's Manual is needed. This manual should be comprised of detailed step-by step (How to Use ...) illustrations for all the CAPS subsystems (user interface, software design database, and execution support system) and all their associated tools that are used in the CAPS software development environment

Much of the work done for this thesis is one approach to developing such essential users documentation. It will be very useful since this thesis addresses only one subsystem and one of its associated tools. In addition to the visual and graphical illustrations, a CAPS development environment glossary is included, it provides the listing of some of the most common used terms and each term is followed by its meaning. Also included are some X-Windows and Microsoft Windows graphical user interface terminology.



## LIST OF REFERENCES

1. Luqi, *Software Evolution Through Rapid Prototyping*, IEEE Computer, pp. 13-25, May, 1989.
2. Luqi, V. Berzins, R. Yeh, "A Prototyping Language for Real-Time Software," *IEEE Transactions on Software Engineering*, October 1988, Vol. 14, No. 10, pp. 1409-1423.
3. Luqi, "Computer-Aided Prototyping for a Command-and-Control System Using CAPS," *IEEE Software*, Vol. 9, No. 1, Jan. 1992, pp. 56-67.
4. R. Steigerwald, Luqi, J. McDowell, "A CASE Tool for Reusable Software Component Storage and Retrieval in Rapid Prototyping," *Information and Software Technology*, England, Vol. 38, No. 9, Nov. 1991, pp. 698-706.
5. Luqi, M. Shing, *CAPS - A Tool For Real-Time System Development and Acquisition*, Naval Research Reviews, pp. 12-16, January, 1992.
6. Luqi, M. Shing, "Teaching Hard Real-Time Software Development via Prototyping," *Proceedings of the International Workshop on Software Engineering Education, at the International Conference on Software Engineering*, Sorrento, Italy, May 21, 1994, pp. 199- 211.
7. Luqi, M. Shing, "Real-Scheduling for Software Prototyping," *Journal of Systems Integration*, Vol. 6, No. 1/2, March. 1996, pp. 41-71.
8. D. Dampier, V. Berzins, "Software Merge: Combining Changes to Decompositions," *Journal of Systems Integration*, Vol. 6, No. 1/2, March. 1996, pp. 135-150.
9. J Goguen, D. Nguyen, J. Meseguer, Luqi, D. Zhang, V. Berzins, "Software Component Search," *Journal of Systems Integration*, Vol. 6, No. 1/2, March. 1996, pp. 93-97,120.
10. X. Li, M. Ketabchi, "A Model-Based Computer-Aided Prototyping System," *Journal of Systems Integration*, Vol. 6, No. 1/2, March. 1996, pp. 135-150.

11. Grosenheider, R., *Enhancements for the CAPS Prototyping System Description Language Syntax-Directed Editor*, Master's Thesis, Naval Postgraduate School, Monterey, Ca. March 1996.
12. Hatley D. and Imtiaz A. Pirbhai, *Strategies for Real-Time System Specification*, Dorset House Publishing, New York, New York, 1988. ISBN:0-932633-11-0.
13. V. Berzins, Luqi, *Software Engineering with Abstractions: An Integrated Approach to Software Development using Ada*, Addison-Wesley Publishing Company, Reading, MA, 1991, ISBN 0-201-08004-4.
14. Behforooz A. and F. J. Hudson, *Software Engineering Fundamentals*, Oxford University Press, Inc., New York, New York, 1996. ISBN:0-19510539-7.
15. Mui L., *When You Can't Find Your UNIX System Administrator*, O'Reilly & Associates, Inc., Sebastopol, CA, 1995. ISBN:1-5692-104-6.
16. Margolis P., *Personal Computer Dictionary*, Random House, Inc., 1996, ISBN: 0-679-76424-0.
17. Sun Microsystems, *OpenWindows Version 3.1 DeskSet Reference Guide*, Sun Microsystems, Inc., Mountain View, CA, 1992.
18. Kinkoph Sherry, *WordPerfect 6 for Windows*, Alpha Books, 1993, ISBN: 1-56761-297-0.
19. MGI Software Corp., *MGI PhotoSuite Version 8.0 Idea Guide and Reference Manual*, MGI Software Corp., 1996.
20. Hergert Douglas, *How To Use Windows 95*, Ziff-Davis Press, Emeryville, CA, 1995, ISBN: 1-56276-268-0.
21. Ferguson P., *Motif Reference Manual*, O'Reilly & Associates Inc., 1993.
22. Heller D. and P. Ferguson, *Motif Programming Manual*, O'Reilly & Associates Inc., 1993.
23. "The Illustrated Book of Terms and Technologies," Dictionary, *PC Novice Learning Series*, Winter 1996.

## INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center 8725 John J. Kingman Road, Suite 0944 Fort Belvoir, VA 22060-6218	2
2.	Dudley Knox Library Naval Postgraduate School 411 Dyer Road Monterey, CA 93943	2
3.	Center for Naval Analysis 4401 Ford Avenue Alexandria, VA 22302	1
4.	Dr. Ted Lewis, Chairman, Code CS Computer Science Department Naval Postgraduate School Monterey, CA 93943	1
5.	Chief of Naval Research 800 North Quincy Street Arlington, VA 22217	1
6.	Dr. Luqi, Code CS/Lq Computer Science Department Naval Postgraduate School Monterey, CA 93943	18
7.	LT Antionette Bell P O Box 585 Ellabell, GA 31308	2